short_2

November 10, 2023

[ ]:
```
!pip install autogluon matplotlib
```

1 Config

[14]:
```
# config

label = 'y'
metric = 'mean_absolute_error'
time_limit = None
presets = None#'best_quality'

use_is_estimated_attr = True

num_seeds = 3

drop_night_outliers = True

# to_drop = ["snow_drift:idx", "snow_density:kgm3", "wind_speed_w_1000hPa:ms",␣
 ↪"dew_or_rime:idx", "prob_rime:p", "fresh_snow_12h:cm", "fresh_snow_24h:cm",␣
 ↪"wind_speed_u_10m:ms", "wind_speed_v_10m:ms", "snow_melt_10min:mm",␣
 ↪"rain_water:kgm2", "dew_point_2m:K", "precip_5min:mm", "absolute_humidity_2m:
 ↪gm3", "air_density_2m:kgm3"]#, "msl_pressure:hPa", "pressure_50m:hPa", ␣
 ↪"pressure_100m:hPa"]
to_drop = ["wind_speed_w_1000hPa:ms", "wind_speed_u_10m:ms", "wind_speed_v_10m:
 ↪ms"]


num_stack_levels = 0
num_bag_folds = 4# 8
num_bag_sets = 1#20

use_tune_data = True
use_test_data = False
use_bag_holdout = True # Enable this if there is a large gap between score_val␣
 ↪and score_test in stack models.

clip_predictions = True
```

2 Loading and preprocessing

[15]:
```
import pandas as pd
import numpy as np



import warnings
warnings.filterwarnings("ignore")


def feature_engineering(X):
```

```
    columns = ['clear_sky_energy_1h:J', 'diffuse_rad_1h:J', 'direct_rad_1h:J',
               'fresh_snow_12h:cm', 'fresh_snow_1h:cm', 'fresh_snow_24h:cm',
               'fresh_snow_3h:cm', 'fresh_snow_6h:cm']

    # Filter rows where index.minute == 0
    X_no_mean = X[X.index.minute == 0][columns].copy()

    date_calc = None
    # If 'date_calc' is present, handle it
    if 'date_calc' in X.columns:
        date_calc = X[X.index.minute == 0]['date_calc']

    X = X.resample('H').mean()

    X[columns] = X_no_mean[columns]

    if date_calc is not None:
        X['date_calc'] = date_calc
    return X




def fix_X(X, name):
    # Convert 'date_forecast' to datetime format and replace original column␣
 ↪with 'ds'
    X['ds'] = pd.to_datetime(X['date_forecast'])
    X.drop(columns=['date_forecast'], inplace=True, errors='ignore')
    X.sort_values(by='ds', inplace=True)
    X.set_index('ds', inplace=True)

    X = feature_engineering(X)

    return X



def handle_features(X_train_observed, X_train_estimated, X_test, y_train):
    X_train_observed = fix_X(X_train_observed, "X_train_observed")
    X_train_estimated = fix_X(X_train_estimated, "X_train_estimated")
    X_test = fix_X(X_test, "X_test")

    y_train['ds'] = pd.to_datetime(y_train['time'])
    y_train.drop(columns=['time'], inplace=True)
    y_train.sort_values(by='ds', inplace=True)
    y_train.set_index('ds', inplace=True)
```

```
    return X_train_observed, X_train_estimated, X_test, y_train




def preprocess_data(X_train_observed, X_train_estimated, X_test, y_train,␣
 ↪location):
    # convert to datetime
    X_train_observed, X_train_estimated, X_test, y_train =␣
 ↪handle_features(X_train_observed, X_train_estimated, X_test, y_train)

    if use_is_estimated_attr:
        X_train_observed["is_estimated"] = 0
        X_train_estimated["is_estimated"] = 1
        X_test["is_estimated"] = 1


    # drop date_calc
    X_train_estimated.drop(columns=['date_calc'], inplace=True)
    X_test.drop(columns=['date_calc'], inplace=True)


    y_train["y"] = y_train["pv_measurement"].astype('float64')
    y_train.drop(columns=['pv_measurement'], inplace=True)
    X_train = pd.concat([X_train_observed, X_train_estimated])


    # clip all y values to 0 if negative
    y_train["y"] = y_train["y"].clip(lower=0)

    X_train = pd.merge(X_train, y_train, how="inner", left_index=True,␣
 ↪right_index=True)

    # print number of nans in y
    print(f"Number of nans in y: {X_train['y'].isna().sum()}")

    print(f"Size of estimated after dropping nans:␣
 ↪{len(X_train[X_train['is_estimated']==1].dropna(subset=['y']))}")


    X_train["location"] = location
    X_test["location"] = location

    return X_train, X_test
# Define locations
locations = ['A', 'B', 'C']
```

```
X_trains = []
X_tests = []
# Loop through locations
for loc in locations:
    print(f"Processing location {loc}...")
    # Read target training data
    y_train = pd.read_parquet(f'{loc}/train_targets.parquet')

    # Read estimated training data and add location feature
    X_train_estimated = pd.read_parquet(f'{loc}/X_train_estimated.parquet')

    # Read observed training data and add location feature
    X_train_observed= pd.read_parquet(f'{loc}/X_train_observed.parquet')

    # Read estimated test data and add location feature
    X_test_estimated = pd.read_parquet(f'{loc}/X_test_estimated.parquet')

    # Preprocess data
    X_train, X_test = preprocess_data(X_train_observed, X_train_estimated,␣
 ↪X_test_estimated, y_train, loc)

    X_trains.append(X_train)
    X_tests.append(X_test)

# Concatenate all data and save to csv
X_train = pd.concat(X_trains)
X_test = pd.concat(X_tests)
```

```
Processing location A…
Number of nans in y: 0
Size of estimated after dropping nans: 4418
Processing location B…
Number of nans in y: 4
Size of estimated after dropping nans: 3625
Processing location C…
Number of nans in y: 6059
Size of estimated after dropping nans: 2954
```

2.1 Feature enginering

2.1.1 Remove anomalies

[16]:
```
def replace_streaks_with_nan(df, max_streak_length, column="y"):
    for location in df["location"].unique():
        x = df[df["location"] == location][column].copy()

        last_val = None
        streak_length = 1
```

```
        streak_indices = []
        allowed = [0]
        found_streaks = {}

        for idx in x.index:
            value = x[idx]
            if value == last_val and value not in allowed:
                streak_length += 1
                streak_indices.append(idx)
            else:
                streak_length = 1
                last_val = value
                streak_indices.clear()

            if streak_length > max_streak_length:
                found_streaks[value] = streak_length

                for streak_idx in streak_indices:
                    x[idx] = np.nan
                streak_indices.clear()  # clear after setting to NaN to avoid␣
 ↪setting multiple times
        df.loc[df["location"] == location, column] = x

        print(f"Found streaks for location {location}: {found_streaks}")

    return df


X_train = replace_streaks_with_nan(X_train.copy(), 3, "y")
```

```
Found streaks for location A: {}
Found streaks for location B: {3.45: 28, 6.9: 7, 12.9375: 5, 13.8: 8, 276.0: 78,
18.975: 58, 0.8625: 4, 118.1625: 33, 34.5: 11, 183.7125: 1058, 87.1125: 7,
79.35: 34, 7.7625: 12, 27.6: 448, 273.41249999999997: 72, 264.78749999999997:
55, 169.05: 33, 375.1875: 56, 314.8125: 66, 76.7625: 10, 135.4125: 216, 81.9375:
202, 2.5875: 12, 81.075: 210}
Found streaks for location C: {9.8: 4, 29.400000000000002: 4, 19.6: 4}
```

[17]:
```
# print num rows
temprows = len(X_train)
X_train.dropna(subset=['y', 'direct_rad_1h:J', 'diffuse_rad_1h:J'],␣
 ↪inplace=True)
print("Dropped rows: ", temprows - len(X_train))
```

```
Dropped rows:  9285
```

[18]:
```
thresh = 0.1
```

```
mask = (X_train["direct_rad_1h:J"] <= thresh) & (X_train["diffuse_rad_1h:J"] <=␣
 ↪thresh) & (X_train["y"] >= 0.1)
if drop_night_outliers:
    X_train.loc[mask, "y"] = np.nan
```

[19]:
```
# print num rows
temprows = len(X_train)
X_train.dropna(subset=['y', 'direct_rad_1h:J', 'diffuse_rad_1h:J'],␣
 ↪inplace=True)
print("Dropped rows: ", temprows - len(X_train))
```

```
Dropped rows:  1876
```

[20]:
```
X_train.drop(columns=to_drop, inplace=True)
X_test.drop(columns=to_drop, inplace=True)

X_train.to_csv('X_train_raw.csv', index=True)
X_test.to_csv('X_test_raw.csv', index=True)
```

[21]:
```
from sklearn.model_selection import train_test_split
def strat_split(x, test_size=0.2, seed=42):
    # create stratified column, location, week, y (mapped to boolean, 0 or␣
 ↪bigger)
    strat = x["location"] + "_" + x["ds"].dt.week.astype('str') + "_" + x["y"].
 ↪apply(lambda x: 0 if x == 0 else 1).astype('str')
    #print(x["strat"])
    print(f"Number of unique strats: {len(strat.unique())}")
    print(f"Lengt of strat: {len(strat)}")

    # check if there are any strats with only one row
    for s in strat.unique():
        if len(x[strat == s]) == 1:
            print(f"Strat {s} has only one row")
            # set equal to another strat with more rows
            strat[strat == s] = strat.unique()[0]

    # split
    train, test = train_test_split(x, test_size=test_size, random_state=seed,␣
 ↪stratify=strat)



    return train, test
```

[22]:
```
from autogluon.tabular import TabularDataset, TabularPredictor
data = TabularDataset('X_train_raw.csv')
data['ds'] = pd.to_datetime(data['ds'])
data = data.sort_values(by='ds')
```

```
split_time = pd.to_datetime("2022-10-28 22:00:00")
train_set = TabularDataset(data[data["ds"] < split_time])
estimated_set = TabularDataset(data[data["ds"] >= split_time]) # only estimated

test_sets = [pd.DataFrame()]*num_seeds
tune_sets = [pd.DataFrame()]*num_seeds
new_train_sets = [pd.DataFrame()]*num_seeds


for location in locations:
    loc_data = data[data["location"] == location]
    num_train_rows = len(loc_data)

    tune_rows = 1500.0 # 2500.0
    if use_test_data:
        tune_rows = 1880.0#max(3000.0,␣
 ↪len(estimated_set[estimated_set["location"] == location]))

    # 3 different seeds
    for i in range(num_seeds):
        holdout_frac = max(0.01, min(0.1, tune_rows / num_train_rows)) *␣
 ↪num_train_rows / len(estimated_set[estimated_set["location"] == location])

        loc_new_train_set, loc_tune_set =␣
 ↪strat_split(estimated_set[estimated_set["location"] == location],␣
 ↪holdout_frac, seed=i)
        new_train_sets[i] = pd.concat([new_train_sets[i], loc_new_train_set])

        if use_test_data:
            loc_tune_set, loc_test_set = strat_split(loc_tune_set, 0.2,␣
 ↪seed=i+1)
            test_sets[i] = pd.concat([test_sets[i], loc_test_set])

        tune_sets[i] = pd.concat([tune_sets[i], loc_tune_set])


train_data_list = [pd.concat([train_set, new_train_set]) for new_train_set in␣
 ↪new_train_sets]
tuning_data_list = tune_sets

test_data_list = [pd.DataFrame()]*num_seeds
if use_test_data:
    test_data_list = test_sets
    for i in range(num_seeds):
        print("Shape of test", test_data_list[i].shape[0])
```

```
train_data_list = [TabularDataset(train_data) for train_data in train_data_list]
tuning_data_list = [TabularDataset(tuning_data) for tuning_data in␣
 ↪tuning_data_list]

if use_test_data:
    test_data_list = [TabularDataset(test_data) for test_data in test_data_list]
```

```
Loaded data from: X_train_raw.csv | Columns = 46 / 46 | Rows = 87925 -> 87925
```

```
Number of unique strats: 54
Lengt of strat: 4216
Number of unique strats: 54
Lengt of strat: 4216
Number of unique strats: 54
Lengt of strat: 4216
Number of unique strats: 49
Lengt of strat: 3535
Number of unique strats: 49
Lengt of strat: 3535
Number of unique strats: 49
Lengt of strat: 3535
Number of unique strats: 46
Lengt of strat: 2925
Number of unique strats: 46
Lengt of strat: 2925
Number of unique strats: 46
Lengt of strat: 2925
```

3 Modeling

[23]:
```
import os

# if submissions folder does not exist, create it
if not os.path.exists('submissions'):
    os.makedirs('submissions')

# Get the last submission number
last_submission_number = int(max([int(filename.split('_')[1].split('.')[0]) for␣
 ↪filename in os.listdir('submissions') if "submission" in filename]))
print("Last submission number:", last_submission_number)
print("Now creating submission number:", last_submission_number + 1)

# Create the new filename
new_filename = f'submission_{last_submission_number + 1}'

print("New filename:", new_filename)
```

```
Last submission number: 132
Now creating submission number: 133
New filename: submission_133
```

[24]:
```
predictors = [None, None, None]
```

[25]:
```
# to mean ensemble different seeds
class NaiveEnsemble:
    def __init__(self, predictors):
        self.predictors = predictors

    def predict(self, x):
        predictions = []
        for predictor in self.predictors:
            predictions.append(predictor.predict(x))
        return np.mean(predictions, axis=0)
```

[26]:
```
from autogluon.common import space
def fit_predictor_for_location(loc):
    different_seeds_predictors = []
    for i, (train_data, tuning_data, test_data) in␣
 ↪enumerate(zip(train_data_list, tuning_data_list, test_data_list)):
        print(f"Training model for location {loc}, seed {i}...")

        hyperparameters = {
            'NN_TORCH': {},
            'XT': [{'criterion': 'squared_error', 'ag_args': {'name_suffix':␣
 ↪'MSE', 'problem_types': ['regression', 'quantile']}}],
            'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}}],#,␣
 ↪{'extra_trees': True, 'feature_fraction': 0.7832570544199176,␣
 ↪'learning_rate': 0.021720607471727896, 'min_data_in_leaf': 3, 'num_leaves':␣
 ↪21, 'ag_args': {'name_suffix': '_r118', 'priority': 17}}],
            'FASTAI': {},
            'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},␣
 ↪{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
        }



        predictor = TabularPredictor(
            label=label,
            eval_metric=metric,
            path=f"AutogluonModels/{new_filename}_{loc}_seed_{i}",
        ).fit(
            train_data=train_data[train_data["location"] == loc].
 ↪reset_index(drop=True).drop(columns=["ds"]),
            time_limit=time_limit,
```

```
            presets=presets,
            num_stack_levels=num_stack_levels,
            num_bag_folds=num_bag_folds,
            num_bag_sets=num_bag_sets,
            tuning_data=tuning_data[tuning_data["location"] == loc].
 ↪reset_index(drop=True).drop(columns=["ds"]),
            use_bag_holdout=use_bag_holdout,
            hyperparameters=hyperparameters,
            #hyperparameter_tune_kwargs=hyperparameter_tune_kwargs,
            excluded_model_types=["XT"] if loc == "C" else ["KNN"] if loc=="B"␣
 ↪else ["KNN", "XT"]
        )

        # evaluate on test data
        if use_test_data:
            t = test_data[test_data["location"] == loc]
            perf = predictor.evaluate(t)
            print("Evaluation on test data:")
            print(perf[predictor.eval_metric.name])

        different_seeds_predictors.append(predictor)

    return NaiveEnsemble(different_seeds_predictors)

loc = "A"
predictors[0] = fit_predictor_for_location(loc)
```

```
Warning: path already exists! This predictor may overwrite an existing
predictor! path="AutogluonModels/submission_133_A_seed_0"
Beginning AutoGluon training …
AutoGluon will save models to "AutogluonModels/submission_133_A_seed_0/"
AutoGluon Version:  0.8.2
Python Version:     3.10.12
Operating System:   Darwin
Platform Machine:   arm64
Platform Version:   Darwin Kernel Version 22.1.0: Sun Oct  9 20:15:09 PDT 2022;
root:xnu-8792.41.9~2/RELEASE_ARM64_T6000
Disk Space Avail:   113.64 GB / 494.38 GB (23.0%)
Train Data Rows:    31283
Train Data Columns: 44
Tuning Data Rows:    1500
Tuning Data Columns: 44
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and many unique label-values observed).
        Label info (max, min, mean, stddev): (5733.42, 0.0, 668.96033,
```

```
1193.25339)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    3636.12 MB
        Train Data (Original)  Memory Usage: 13.18 MB (0.4% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.

Training model for location A, seed 0…

        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 2 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 3): ['elevation:m', 'snow_drift:idx',
'location']
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 40 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])   :  1 | ['is_estimated']
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 39 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', ['bool']) :  2 | ['snow_density:kgm3', 'is_estimated']
        0.1s = Fit runtime
        41 features in original data used to generate 41 features in processed
data.
        Train Data (Processed) Memory Usage: 10.29 MB (0.3% of available memory)
Data preprocessing and feature engineering runtime = 0.24s …
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
```

```
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
use_bag_holdout=True, will use tuning_data as holdout (will not be used for
early stopping).
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'XT': [{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}},
{'extra_trees': True, 'feature_fraction': 0.7832570544199176, 'learning_rate':
0.021720607471727896, 'min_data_in_leaf': 3, 'num_leaves': 21, 'ag_args':
{'name_suffix': '_r118', 'priority': 17}}],
        'FASTAI': {},
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
Excluded models: ['XT', 'KNN'] (Specified by `excluded_model_types`)
Fitting 4 L1 models …
Fitting model: LightGBMXT_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy

[1000]  valid_set's l1: 186.795
[2000]  valid_set's l1: 179.852
[3000]  valid_set's l1: 176.284
[4000]  valid_set's l1: 174.52
[5000]  valid_set's l1: 173.313
[6000]  valid_set's l1: 172.465
[7000]  valid_set's l1: 171.922
[8000]  valid_set's l1: 171.46
[9000]  valid_set's l1: 171.077
[10000] valid_set's l1: 170.846
[1000]  valid_set's l1: 179.581
[2000]  valid_set's l1: 174.837
[3000]  valid_set's l1: 172.123
[4000]  valid_set's l1: 170.34
[5000]  valid_set's l1: 169.519
[6000]  valid_set's l1: 169.152
[7000]  valid_set's l1: 168.552
[8000]  valid_set's l1: 168.137
[9000]  valid_set's l1: 167.78
[10000] valid_set's l1: 167.564
[1000]  valid_set's l1: 184.43
[2000]  valid_set's l1: 178.208
[3000]  valid_set's l1: 175.162
[4000]  valid_set's l1: 172.96
```

```
[5000]  valid_set's l1: 171.722
[6000]  valid_set's l1: 171.075
[7000]  valid_set's l1: 170.625
[8000]  valid_set's l1: 170.051
[9000]  valid_set's l1: 169.724
[10000] valid_set's l1: 169.322
[1000]  valid_set's l1: 187.138
[2000]  valid_set's l1: 181.284
[3000]  valid_set's l1: 179.016
[4000]  valid_set's l1: 177.541
[5000]  valid_set's l1: 176.458
[6000]  valid_set's l1: 175.666
[7000]  valid_set's l1: 175.09
[8000]  valid_set's l1: 174.441
[9000]  valid_set's l1: 174.164
[10000] valid_set's l1: 173.904
[1000]  valid_set's l1: 179.629
[2000]  valid_set's l1: 176.018
[3000]  valid_set's l1: 174.08
[4000]  valid_set's l1: 172.48
[5000]  valid_set's l1: 171.686
[6000]  valid_set's l1: 170.721
[7000]  valid_set's l1: 170.159
[8000]  valid_set's l1: 169.955
[9000]  valid_set's l1: 169.767
[10000] valid_set's l1: 169.575
[1000]  valid_set's l1: 185.209
[2000]  valid_set's l1: 179.793
[3000]  valid_set's l1: 177.331
[4000]  valid_set's l1: 175.801
[5000]  valid_set's l1: 174.663
[6000]  valid_set's l1: 173.824
[7000]  valid_set's l1: 173.113
[8000]  valid_set's l1: 172.713
[9000]  valid_set's l1: 172.34
[10000] valid_set's l1: 172.059
[1000]  valid_set's l1: 181.203
[2000]  valid_set's l1: 174.787
[3000]  valid_set's l1: 172.307
[4000]  valid_set's l1: 170.974
[5000]  valid_set's l1: 169.904
[6000]  valid_set's l1: 169.205
[7000]  valid_set's l1: 168.792
[8000]  valid_set's l1: 168.304
[9000]  valid_set's l1: 167.964
[10000] valid_set's l1: 167.732
[1000]  valid_set's l1: 184.334
[2000]  valid_set's l1: 179.976
```

```
[3000]  valid_set's l1: 177.315
[4000]  valid_set's l1: 175.727
[5000]  valid_set's l1: 174.763
[6000]  valid_set's l1: 174.04
[7000]  valid_set's l1: 173.499
[8000]  valid_set's l1: 173.121
[9000]  valid_set's l1: 172.805
[10000] valid_set's l1: 172.455
[1000]  valid_set's l1: 183.383
[2000]  valid_set's l1: 177.711
[3000]  valid_set's l1: 174.658
[4000]  valid_set's l1: 172.708
[5000]  valid_set's l1: 171.808
[6000]  valid_set's l1: 170.969
[7000]  valid_set's l1: 170.424
[8000]  valid_set's l1: 170.113
[9000]  valid_set's l1: 169.873
[10000] valid_set's l1: 169.601
[1000]  valid_set's l1: 190.193
[2000]  valid_set's l1: 183.264
[3000]  valid_set's l1: 179.794
[4000]  valid_set's l1: 177.211
[5000]  valid_set's l1: 175.686
[6000]  valid_set's l1: 174.641
[7000]  valid_set's l1: 173.905
[8000]  valid_set's l1: 173.322
[9000]  valid_set's l1: 172.995
[10000] valid_set's l1: 172.721
[1000]  valid_set's l1: 180.168
[2000]  valid_set's l1: 175.147
[3000]  valid_set's l1: 172.384
[4000]  valid_set's l1: 170.532
[5000]  valid_set's l1: 169.126
[6000]  valid_set's l1: 168.416
[7000]  valid_set's l1: 167.837
[8000]  valid_set's l1: 167.404
[9000]  valid_set's l1: 167.051
[10000] valid_set's l1: 166.893
[1000]  valid_set's l1: 179.692
[2000]  valid_set's l1: 173.862
[3000]  valid_set's l1: 171.154
[4000]  valid_set's l1: 169.353
[5000]  valid_set's l1: 168.263
[6000]  valid_set's l1: 167.28
[7000]  valid_set's l1: 166.695
[8000]  valid_set's l1: 166.225
[9000]  valid_set's l1: 165.879
[10000] valid_set's l1: 165.679
```

```
        -88.6573         = Validation score   (-mean_absolute_error)
        1746.71s         = Training   runtime
        14.28s   = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -107.1287        = Validation score   (-mean_absolute_error)
        198.08s  = Training   runtime
        0.42s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -87.9278         = Validation score   (-mean_absolute_error)
        820.77s  = Training   runtime
        0.54s    = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy

[1000]  valid_set's l1: 198.484
[2000]  valid_set's l1: 191.185
[3000]  valid_set's l1: 186.69
[4000]  valid_set's l1: 183.754
[5000]  valid_set's l1: 181.578
[6000]  valid_set's l1: 179.344
[7000]  valid_set's l1: 178.059
[8000]  valid_set's l1: 176.805
[9000]  valid_set's l1: 175.61
[10000] valid_set's l1: 174.724
[1000]  valid_set's l1: 188.805
[2000]  valid_set's l1: 182.26
[3000]  valid_set's l1: 178.535
[4000]  valid_set's l1: 175.971
[5000]  valid_set's l1: 173.822
[6000]  valid_set's l1: 172.26
[7000]  valid_set's l1: 171.035
[8000]  valid_set's l1: 169.937
[9000]  valid_set's l1: 169.057
[10000] valid_set's l1: 168.499
[1000]  valid_set's l1: 191.474
[2000]  valid_set's l1: 185.249
[3000]  valid_set's l1: 181.203
[4000]  valid_set's l1: 178.596
[5000]  valid_set's l1: 176.443
[6000]  valid_set's l1: 174.464
[7000]  valid_set's l1: 173.01
[8000]  valid_set's l1: 171.672
[9000]  valid_set's l1: 170.68
```

```
[10000] valid_set's l1: 169.856
[1000]  valid_set's l1: 198.112
[2000]  valid_set's l1: 191.258
[3000]  valid_set's l1: 186.853
[4000]  valid_set's l1: 183.862
[5000]  valid_set's l1: 181.48
[6000]  valid_set's l1: 179.831
[7000]  valid_set's l1: 178.69
[8000]  valid_set's l1: 177.626
[9000]  valid_set's l1: 176.641
[10000] valid_set's l1: 176.029
[1000]  valid_set's l1: 187.988
[2000]  valid_set's l1: 181.73
[3000]  valid_set's l1: 177.995
[4000]  valid_set's l1: 175.283
[5000]  valid_set's l1: 173.403
[6000]  valid_set's l1: 171.872
[7000]  valid_set's l1: 170.498
[8000]  valid_set's l1: 169.323
[9000]  valid_set's l1: 168.578
[10000] valid_set's l1: 167.851
[1000]  valid_set's l1: 192.854
[2000]  valid_set's l1: 187.088
[3000]  valid_set's l1: 183.26
[4000]  valid_set's l1: 180.281
[5000]  valid_set's l1: 178.244
[6000]  valid_set's l1: 176.536
[7000]  valid_set's l1: 175.18
[8000]  valid_set's l1: 173.935
[9000]  valid_set's l1: 173.011
[10000] valid_set's l1: 172.155
[1000]  valid_set's l1: 190.884
[2000]  valid_set's l1: 183.728
[3000]  valid_set's l1: 179.424
[4000]  valid_set's l1: 176.563
[5000]  valid_set's l1: 174.276
[6000]  valid_set's l1: 172.344
[7000]  valid_set's l1: 170.649
[8000]  valid_set's l1: 169.57
[9000]  valid_set's l1: 168.436
[10000] valid_set's l1: 167.652
[1000]  valid_set's l1: 192.424
[2000]  valid_set's l1: 185.87
[3000]  valid_set's l1: 182.081
[4000]  valid_set's l1: 179.712
[5000]  valid_set's l1: 177.734
[6000]  valid_set's l1: 176.218
[7000]  valid_set's l1: 175.192
```

```
[8000]  valid_set's l1: 174.089
[9000]  valid_set's l1: 173.099
[10000] valid_set's l1: 172.337
[1000]  valid_set's l1: 192.52
[2000]  valid_set's l1: 185.769
[3000]  valid_set's l1: 181.853
[4000]  valid_set's l1: 179.133
[5000]  valid_set's l1: 177.102
[6000]  valid_set's l1: 175.605
[7000]  valid_set's l1: 174.169
[8000]  valid_set's l1: 173.158
[9000]  valid_set's l1: 172.238
[10000] valid_set's l1: 171.436
[1000]  valid_set's l1: 200.559
[2000]  valid_set's l1: 192.996
[3000]  valid_set's l1: 188.598
[4000]  valid_set's l1: 185.047
[5000]  valid_set's l1: 182.323
[6000]  valid_set's l1: 180.509
[7000]  valid_set's l1: 179.016
[8000]  valid_set's l1: 177.598
[9000]  valid_set's l1: 176.546
[10000] valid_set's l1: 175.601
[1000]  valid_set's l1: 191.069
[2000]  valid_set's l1: 183.998
[3000]  valid_set's l1: 179.84
[4000]  valid_set's l1: 176.847
[5000]  valid_set's l1: 174.801
[6000]  valid_set's l1: 173.226
[7000]  valid_set's l1: 171.816
[8000]  valid_set's l1: 170.921
[9000]  valid_set's l1: 169.928
[10000] valid_set's l1: 168.939
[1000]  valid_set's l1: 188.964
[2000]  valid_set's l1: 182.198
[3000]  valid_set's l1: 177.755
[4000]  valid_set's l1: 174.886
[5000]  valid_set's l1: 172.467
[6000]  valid_set's l1: 170.852
[7000]  valid_set's l1: 169.208
[8000]  valid_set's l1: 168.119
[9000]  valid_set's l1: 167.099
[10000] valid_set's l1: 166.288
        -89.8551         = Validation score   (-mean_absolute_error)
        3583.25s         = Training   runtime
        9.19s    = Validation runtime
Fitting model: WeightedEnsemble_L2 …
```

```
        -84.4524         = Validation score   (-mean_absolute_error)
        0.08s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 6395.3s … Best model:
"WeightedEnsemble_L2"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_133_A_seed_0/")
Beginning AutoGluon training …
AutoGluon will save models to "AutogluonModels/submission_133_A_seed_1/"
AutoGluon Version:  0.8.2
Python Version:     3.10.12
Operating System:   Darwin
Platform Machine:   arm64
Platform Version:   Darwin Kernel Version 22.1.0: Sun Oct  9 20:15:09 PDT 2022;
root:xnu-8792.41.9~2/RELEASE_ARM64_T6000
Disk Space Avail:   111.78 GB / 494.38 GB (22.6%)
Train Data Rows:    31283
Train Data Columns: 44
Tuning Data Rows:    1500
Tuning Data Columns: 44
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and many unique label-values observed).
        Label info (max, min, mean, stddev): (5733.42, 0.0, 668.42566,
1192.04707)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    2963.35 MB
        Train Data (Original)  Memory Usage: 13.18 MB (0.4% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.

Training model for location A, seed 1…

        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 2 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
```

```
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 3): ['elevation:m', 'snow_drift:idx',
'location']
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 40 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])   :  1 | ['is_estimated']
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 39 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', ['bool']) :  2 | ['snow_density:kgm3', 'is_estimated']
        0.3s = Fit runtime
        41 features in original data used to generate 41 features in processed
data.
        Train Data (Processed) Memory Usage: 10.29 MB (0.3% of available memory)
Data preprocessing and feature engineering runtime = 0.5s …
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
use_bag_holdout=True, will use tuning_data as holdout (will not be used for
early stopping).
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'XT': [{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}},
{'extra_trees': True, 'feature_fraction': 0.7832570544199176, 'learning_rate':
0.021720607471727896, 'min_data_in_leaf': 3, 'num_leaves': 21, 'ag_args':
{'name_suffix': '_r118', 'priority': 17}}],
        'FASTAI': {},
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
Excluded models: ['XT', 'KNN'] (Specified by `excluded_model_types`)
Fitting 4 L1 models …
Fitting model: LightGBMXT_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
```

```
[1000]  valid_set's l1: 184.987
[2000]  valid_set's l1: 178.462
[3000]  valid_set's l1: 175.58
[4000]  valid_set's l1: 173.262
[5000]  valid_set's l1: 171.609
[6000]  valid_set's l1: 170.441
[7000]  valid_set's l1: 169.716
[8000]  valid_set's l1: 169.361
[9000]  valid_set's l1: 168.817
[10000] valid_set's l1: 168.53
[1000]  valid_set's l1: 180.141
[2000]  valid_set's l1: 174.776
[3000]  valid_set's l1: 172.413
[4000]  valid_set's l1: 171.26
[5000]  valid_set's l1: 170.249
[6000]  valid_set's l1: 169.643
[7000]  valid_set's l1: 169.064
[8000]  valid_set's l1: 168.71
[9000]  valid_set's l1: 168.382
[10000] valid_set's l1: 168.091
[1000]  valid_set's l1: 182.861
[2000]  valid_set's l1: 176.269
[3000]  valid_set's l1: 173.213
[4000]  valid_set's l1: 171.214
[5000]  valid_set's l1: 170.237
[6000]  valid_set's l1: 169.728
[7000]  valid_set's l1: 168.964
[8000]  valid_set's l1: 168.381
[9000]  valid_set's l1: 167.992
[10000] valid_set's l1: 167.67
[1000]  valid_set's l1: 187.744
[2000]  valid_set's l1: 182.45
[3000]  valid_set's l1: 179.692
[4000]  valid_set's l1: 177.789
[5000]  valid_set's l1: 176.816
[6000]  valid_set's l1: 175.912
[7000]  valid_set's l1: 175.459
[8000]  valid_set's l1: 174.906
[9000]  valid_set's l1: 174.599
[10000] valid_set's l1: 174.233
[1000]  valid_set's l1: 178.511
[2000]  valid_set's l1: 174.787
[3000]  valid_set's l1: 172.554
[4000]  valid_set's l1: 171.412
[5000]  valid_set's l1: 170.344
[6000]  valid_set's l1: 169.594
[7000]  valid_set's l1: 169.241
[8000]  valid_set's l1: 168.88
```

```
[9000]  valid_set's l1: 168.628
[10000] valid_set's l1: 168.344
[1000]  valid_set's l1: 187.095
[2000]  valid_set's l1: 182.31
[3000]  valid_set's l1: 180.276
[4000]  valid_set's l1: 178.848
[5000]  valid_set's l1: 177.318
[6000]  valid_set's l1: 176.496
[7000]  valid_set's l1: 176.029
[8000]  valid_set's l1: 175.639
[9000]  valid_set's l1: 175.29
[10000] valid_set's l1: 175.067
[1000]  valid_set's l1: 182.087
[2000]  valid_set's l1: 175.379
[3000]  valid_set's l1: 172.127
[4000]  valid_set's l1: 170.634
[5000]  valid_set's l1: 169.724
[6000]  valid_set's l1: 169.181
[7000]  valid_set's l1: 168.585
[8000]  valid_set's l1: 168.186
[9000]  valid_set's l1: 167.977
[10000] valid_set's l1: 167.692
[1000]  valid_set's l1: 186.8
[2000]  valid_set's l1: 182.648
[3000]  valid_set's l1: 180.51
[4000]  valid_set's l1: 179.462
[5000]  valid_set's l1: 178.237
[6000]  valid_set's l1: 177.453
[7000]  valid_set's l1: 177.048
[8000]  valid_set's l1: 176.767
[9000]  valid_set's l1: 176.366
[10000] valid_set's l1: 176.098
[1000]  valid_set's l1: 183.921
[2000]  valid_set's l1: 179.888
[3000]  valid_set's l1: 177.762
[4000]  valid_set's l1: 175.81
[5000]  valid_set's l1: 174.489
[6000]  valid_set's l1: 173.766
[7000]  valid_set's l1: 173.177
[8000]  valid_set's l1: 172.72
[9000]  valid_set's l1: 172.222
[10000] valid_set's l1: 171.863
[1000]  valid_set's l1: 188.812
[2000]  valid_set's l1: 181.407
[3000]  valid_set's l1: 178.141
[4000]  valid_set's l1: 175.991
[5000]  valid_set's l1: 174.515
[6000]  valid_set's l1: 173.696
```

```
[7000]  valid_set's l1: 173.029
[8000]  valid_set's l1: 172.542
[9000]  valid_set's l1: 172.296
[10000] valid_set's l1: 172.048
[1000]  valid_set's l1: 176.757
[2000]  valid_set's l1: 172.04
[3000]  valid_set's l1: 169.708
[4000]  valid_set's l1: 167.86
[5000]  valid_set's l1: 166.577
[6000]  valid_set's l1: 165.851
[7000]  valid_set's l1: 165.505
[8000]  valid_set's l1: 165.057
[9000]  valid_set's l1: 164.855
[10000] valid_set's l1: 164.61
[1000]  valid_set's l1: 182.421
[2000]  valid_set's l1: 177.191
[3000]  valid_set's l1: 174.651
[4000]  valid_set's l1: 173.161
[5000]  valid_set's l1: 171.892
[6000]  valid_set's l1: 171.29
[7000]  valid_set's l1: 170.922
[8000]  valid_set's l1: 170.552
[9000]  valid_set's l1: 170.268
[10000] valid_set's l1: 169.996

        -86.2485         = Validation score   (-mean_absolute_error)
        2269.65s         = Training   runtime
        13.62s   = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -102.0382        = Validation score   (-mean_absolute_error)
        190.7s   = Training   runtime
        0.43s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -88.2755         = Validation score   (-mean_absolute_error)
        778.93s  = Training   runtime
        0.35s    = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy

[1000]  valid_set's l1: 195.382
[2000]  valid_set's l1: 187.626
[3000]  valid_set's l1: 183.371
[4000]  valid_set's l1: 180.123
[5000]  valid_set's l1: 177.526
```

```
[6000]  valid_set's l1: 175.638
[7000]  valid_set's l1: 174.269
[8000]  valid_set's l1: 173.094
[9000]  valid_set's l1: 172.019
[10000] valid_set's l1: 171.218
[1000]  valid_set's l1: 189.466
[2000]  valid_set's l1: 182.722
[3000]  valid_set's l1: 178.927
[4000]  valid_set's l1: 176.268
[5000]  valid_set's l1: 174.162
[6000]  valid_set's l1: 172.422
[7000]  valid_set's l1: 170.976
[8000]  valid_set's l1: 170.024
[9000]  valid_set's l1: 169.026
[10000] valid_set's l1: 168.264
[1000]  valid_set's l1: 191.407
[2000]  valid_set's l1: 185.067
[3000]  valid_set's l1: 180.989
[4000]  valid_set's l1: 178.047
[5000]  valid_set's l1: 175.756
[6000]  valid_set's l1: 174.048
[7000]  valid_set's l1: 172.732
[8000]  valid_set's l1: 171.547
[9000]  valid_set's l1: 170.61
[10000] valid_set's l1: 169.942
[1000]  valid_set's l1: 198.89
[2000]  valid_set's l1: 191.802
[3000]  valid_set's l1: 187.288
[4000]  valid_set's l1: 184.51
[5000]  valid_set's l1: 182.42
[6000]  valid_set's l1: 180.741
[7000]  valid_set's l1: 179.274
[8000]  valid_set's l1: 178.13
[9000]  valid_set's l1: 177.282
[10000] valid_set's l1: 176.419
[1000]  valid_set's l1: 188.001
[2000]  valid_set's l1: 181.548
[3000]  valid_set's l1: 177.783
[4000]  valid_set's l1: 175.125
[5000]  valid_set's l1: 173.423
[6000]  valid_set's l1: 171.959
[7000]  valid_set's l1: 170.743
[8000]  valid_set's l1: 169.563
[9000]  valid_set's l1: 168.791
[10000] valid_set's l1: 168.04
[1000]  valid_set's l1: 194.767
[2000]  valid_set's l1: 188.551
[3000]  valid_set's l1: 184.893
```

```
[4000]  valid_set's l1: 182.15
[5000]  valid_set's l1: 180.349
[6000]  valid_set's l1: 178.64
[7000]  valid_set's l1: 177.174
[8000]  valid_set's l1: 175.987
[9000]  valid_set's l1: 175.147
[10000] valid_set's l1: 174.356
[1000]  valid_set's l1: 192.214
[2000]  valid_set's l1: 185.089
[3000]  valid_set's l1: 180.635
[4000]  valid_set's l1: 177.234
[5000]  valid_set's l1: 175.042
[6000]  valid_set's l1: 172.902
[7000]  valid_set's l1: 171.53
[8000]  valid_set's l1: 170.345
[9000]  valid_set's l1: 169.377
[10000] valid_set's l1: 168.359
[1000]  valid_set's l1: 195.737
[2000]  valid_set's l1: 188.673
[3000]  valid_set's l1: 185.283
[4000]  valid_set's l1: 182.229
[5000]  valid_set's l1: 180.247
[6000]  valid_set's l1: 178.701
[7000]  valid_set's l1: 177.311
[8000]  valid_set's l1: 176.129
[9000]  valid_set's l1: 175.128
[10000] valid_set's l1: 174.193
[1000]  valid_set's l1: 192.12
[2000]  valid_set's l1: 186.043
[3000]  valid_set's l1: 182.236
[4000]  valid_set's l1: 179.558
[5000]  valid_set's l1: 177.436
[6000]  valid_set's l1: 175.942
[7000]  valid_set's l1: 174.667
[8000]  valid_set's l1: 173.427
[9000]  valid_set's l1: 172.557
[10000] valid_set's l1: 171.961
[1000]  valid_set's l1: 199.522
[2000]  valid_set's l1: 192.279
[3000]  valid_set's l1: 187.63
[4000]  valid_set's l1: 184.472
[5000]  valid_set's l1: 181.671
[6000]  valid_set's l1: 179.729
[7000]  valid_set's l1: 177.881
[8000]  valid_set's l1: 176.337
[9000]  valid_set's l1: 175.114
[10000] valid_set's l1: 174.18
[1000]  valid_set's l1: 188.154
```

```
[2000]  valid_set's l1: 180.984
[3000]  valid_set's l1: 176.695
[4000]  valid_set's l1: 173.665
[5000]  valid_set's l1: 171.857
[6000]  valid_set's l1: 170.457
[7000]  valid_set's l1: 169.163
[8000]  valid_set's l1: 168.135
[9000]  valid_set's l1: 167.086
[10000] valid_set's l1: 166.304
[1000]  valid_set's l1: 191.292
[2000]  valid_set's l1: 183.872
[3000]  valid_set's l1: 179.765
[4000]  valid_set's l1: 176.748
[5000]  valid_set's l1: 174.475
[6000]  valid_set's l1: 172.668
[7000]  valid_set's l1: 171.373
[8000]  valid_set's l1: 170.195
[9000]  valid_set's l1: 169.157
[10000] valid_set's l1: 168.383
        -86.8658         = Validation score   (-mean_absolute_error)
        962.98s  = Training   runtime
        8.08s    = Validation runtime
Fitting model: WeightedEnsemble_L2 …
        -82.8865         = Validation score   (-mean_absolute_error)
        0.04s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 4242.78s … Best model:
"WeightedEnsemble_L2"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_133_A_seed_1/")
Beginning AutoGluon training …
AutoGluon will save models to "AutogluonModels/submission_133_A_seed_2/"
AutoGluon Version:  0.8.2
Python Version:     3.10.12
Operating System:   Darwin
Platform Machine:   arm64
Platform Version:   Darwin Kernel Version 22.1.0: Sun Oct  9 20:15:09 PDT 2022;
root:xnu-8792.41.9~2/RELEASE_ARM64_T6000
Disk Space Avail:   111.10 GB / 494.38 GB (22.5%)
Train Data Rows:    31283
Train Data Columns: 44
Tuning Data Rows:    1500
Tuning Data Columns: 44
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and many unique label-values observed).
```

```
        Label info (max, min, mean, stddev): (5733.42, 0.0, 668.82758,
1191.55609)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    3525.65 MB
        Train Data (Original)  Memory Usage: 13.18 MB (0.4% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 2 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…

Training model for location A, seed 2…

        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 3): ['elevation:m', 'snow_drift:idx',
'location']
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 40 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])   :  1 | ['is_estimated']
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 39 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', ['bool']) :  2 | ['snow_density:kgm3', 'is_estimated']
        0.1s = Fit runtime
        41 features in original data used to generate 41 features in processed
data.
        Train Data (Processed) Memory Usage: 10.29 MB (0.3% of available memory)
Data preprocessing and feature engineering runtime = 0.16s …
AutoGluon will gauge predictive performance using evaluation metric:
```

```
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
use_bag_holdout=True, will use tuning_data as holdout (will not be used for
early stopping).
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'XT': [{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}},
{'extra_trees': True, 'feature_fraction': 0.7832570544199176, 'learning_rate':
0.021720607471727896, 'min_data_in_leaf': 3, 'num_leaves': 21, 'ag_args':
{'name_suffix': '_r118', 'priority': 17}}],
        'FASTAI': {},
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
Excluded models: ['XT', 'KNN'] (Specified by `excluded_model_types`)
Fitting 4 L1 models …
Fitting model: LightGBMXT_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy

[1000]  valid_set's l1: 188.456
[2000]  valid_set's l1: 181.51
[3000]  valid_set's l1: 178.463
[4000]  valid_set's l1: 176.536
[5000]  valid_set's l1: 175.28
[6000]  valid_set's l1: 174.383
[7000]  valid_set's l1: 173.562
[8000]  valid_set's l1: 173.165
[9000]  valid_set's l1: 172.822
[10000] valid_set's l1: 172.557
[1000]  valid_set's l1: 179.725
[2000]  valid_set's l1: 175.049
[3000]  valid_set's l1: 173.027
[4000]  valid_set's l1: 171.213
[5000]  valid_set's l1: 170.198
[6000]  valid_set's l1: 169.243
[7000]  valid_set's l1: 168.726
[8000]  valid_set's l1: 168.227
[9000]  valid_set's l1: 168.065
[10000] valid_set's l1: 167.911
[1000]  valid_set's l1: 181.076
[2000]  valid_set's l1: 175.177
[3000]  valid_set's l1: 172.462
```

```
[4000]  valid_set's l1: 170.788
[5000]  valid_set's l1: 169.745
[6000]  valid_set's l1: 169.029
[7000]  valid_set's l1: 168.549
[8000]  valid_set's l1: 168.042
[9000]  valid_set's l1: 167.71
[10000] valid_set's l1: 167.281
[1000]  valid_set's l1: 186.161
[2000]  valid_set's l1: 180.938
[3000]  valid_set's l1: 177.762
[4000]  valid_set's l1: 176.285
[5000]  valid_set's l1: 174.96
[6000]  valid_set's l1: 174.012
[7000]  valid_set's l1: 173.435
[8000]  valid_set's l1: 172.942
[9000]  valid_set's l1: 172.537
[10000] valid_set's l1: 172.425
[1000]  valid_set's l1: 179.156
[2000]  valid_set's l1: 175.329
[3000]  valid_set's l1: 172.863
[4000]  valid_set's l1: 171.649
[5000]  valid_set's l1: 170.758
[6000]  valid_set's l1: 170.171
[7000]  valid_set's l1: 169.721
[8000]  valid_set's l1: 169.521
[9000]  valid_set's l1: 169.193
[10000] valid_set's l1: 169.102
[1000]  valid_set's l1: 187.637
[2000]  valid_set's l1: 182.727
[3000]  valid_set's l1: 179.89
[4000]  valid_set's l1: 178.191
[5000]  valid_set's l1: 176.986
[6000]  valid_set's l1: 176.326
[7000]  valid_set's l1: 175.873
[8000]  valid_set's l1: 175.29
[9000]  valid_set's l1: 174.938
[10000] valid_set's l1: 174.588
[1000]  valid_set's l1: 182.448
[2000]  valid_set's l1: 175.964
[3000]  valid_set's l1: 172.998
[4000]  valid_set's l1: 171.534
[5000]  valid_set's l1: 170.555
[6000]  valid_set's l1: 169.902
[7000]  valid_set's l1: 169.512
[8000]  valid_set's l1: 169.095
[9000]  valid_set's l1: 168.785
[10000] valid_set's l1: 168.508
[1000]  valid_set's l1: 186.995
```

```
[2000]  valid_set's l1: 182.14
[3000]  valid_set's l1: 179.706
[4000]  valid_set's l1: 178.112
[5000]  valid_set's l1: 177.327
[6000]  valid_set's l1: 176.454
[7000]  valid_set's l1: 175.928
[8000]  valid_set's l1: 175.584
[9000]  valid_set's l1: 175.268
[10000] valid_set's l1: 175.005
[1000]  valid_set's l1: 183.086
[2000]  valid_set's l1: 178.082
[3000]  valid_set's l1: 175.015
[4000]  valid_set's l1: 173.608
[5000]  valid_set's l1: 172.717
[6000]  valid_set's l1: 171.899
[7000]  valid_set's l1: 171.434
[8000]  valid_set's l1: 171.215
[9000]  valid_set's l1: 170.884
[10000] valid_set's l1: 170.581
[1000]  valid_set's l1: 188.158
[2000]  valid_set's l1: 181.096
[3000]  valid_set's l1: 177.281
[4000]  valid_set's l1: 175.247
[5000]  valid_set's l1: 173.663
[6000]  valid_set's l1: 172.654
[7000]  valid_set's l1: 172.202
[8000]  valid_set's l1: 171.71
[9000]  valid_set's l1: 171.351
[10000] valid_set's l1: 171.012
[1000]  valid_set's l1: 176.42
[2000]  valid_set's l1: 171.594
[3000]  valid_set's l1: 168.949
[4000]  valid_set's l1: 167.317
[5000]  valid_set's l1: 166.44
[6000]  valid_set's l1: 165.879
[7000]  valid_set's l1: 165.291
[8000]  valid_set's l1: 164.788
[9000]  valid_set's l1: 164.49
[10000] valid_set's l1: 164.222
[1000]  valid_set's l1: 180.619
[2000]  valid_set's l1: 175.654
[3000]  valid_set's l1: 173.22
[4000]  valid_set's l1: 171.828
[5000]  valid_set's l1: 170.7
[6000]  valid_set's l1: 170.154
[7000]  valid_set's l1: 169.698
[8000]  valid_set's l1: 169.317
[9000]  valid_set's l1: 168.957
```

```
[10000] valid_set's l1: 168.765

        -82.0977         = Validation score   (-mean_absolute_error)
        1287.29s         = Training   runtime
        12.53s   = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -103.7315        = Validation score   (-mean_absolute_error)
        367.78s  = Training   runtime
        0.62s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -93.8811         = Validation score   (-mean_absolute_error)
        719.36s  = Training   runtime
        0.44s    = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy

[1000]  valid_set's l1: 200.411
[2000]  valid_set's l1: 193.014
[3000]  valid_set's l1: 188.201
[4000]  valid_set's l1: 184.457
[5000]  valid_set's l1: 182.066
[6000]  valid_set's l1: 180.032
[7000]  valid_set's l1: 178.296
[8000]  valid_set's l1: 176.87
[9000]  valid_set's l1: 175.571
[10000] valid_set's l1: 174.554
[1000]  valid_set's l1: 188.005
[2000]  valid_set's l1: 182.088
[3000]  valid_set's l1: 178.515
[4000]  valid_set's l1: 175.677
[5000]  valid_set's l1: 173.811
[6000]  valid_set's l1: 172.235
[7000]  valid_set's l1: 171.058
[8000]  valid_set's l1: 170.171
[9000]  valid_set's l1: 169.229
[10000] valid_set's l1: 168.461
[1000]  valid_set's l1: 190.067
[2000]  valid_set's l1: 183.806
[3000]  valid_set's l1: 179.255
[4000]  valid_set's l1: 175.97
[5000]  valid_set's l1: 173.675
[6000]  valid_set's l1: 171.882
[7000]  valid_set's l1: 170.505
[8000]  valid_set's l1: 169.302
```

```
[9000]  valid_set's l1: 168.52
[10000] valid_set's l1: 167.78
[1000]  valid_set's l1: 197.425
[2000]  valid_set's l1: 189.936
[3000]  valid_set's l1: 185.678
[4000]  valid_set's l1: 183.12
[5000]  valid_set's l1: 180.706
[6000]  valid_set's l1: 179.136
[7000]  valid_set's l1: 177.759
[8000]  valid_set's l1: 176.733
[9000]  valid_set's l1: 175.735
[10000] valid_set's l1: 175.046
[1000]  valid_set's l1: 187.788
[2000]  valid_set's l1: 181.62
[3000]  valid_set's l1: 177.69
[4000]  valid_set's l1: 175.287
[5000]  valid_set's l1: 173.76
[6000]  valid_set's l1: 172.361
[7000]  valid_set's l1: 171.197
[8000]  valid_set's l1: 170.086
[9000]  valid_set's l1: 169.335
[10000] valid_set's l1: 168.651
[1000]  valid_set's l1: 195.085
[2000]  valid_set's l1: 189.473
[3000]  valid_set's l1: 185.816
[4000]  valid_set's l1: 182.914
[5000]  valid_set's l1: 180.725
[6000]  valid_set's l1: 178.628
[7000]  valid_set's l1: 177.198
[8000]  valid_set's l1: 176.096
[9000]  valid_set's l1: 175.215
[10000] valid_set's l1: 174.601
[1000]  valid_set's l1: 193.831
[2000]  valid_set's l1: 186.887
[3000]  valid_set's l1: 182.562
[4000]  valid_set's l1: 179.36
[5000]  valid_set's l1: 177.102
[6000]  valid_set's l1: 175.263
[7000]  valid_set's l1: 173.8
[8000]  valid_set's l1: 172.418
[9000]  valid_set's l1: 171.232
[10000] valid_set's l1: 170.319
[1000]  valid_set's l1: 196.864
[2000]  valid_set's l1: 190.195
[3000]  valid_set's l1: 186.451
[4000]  valid_set's l1: 183.981
[5000]  valid_set's l1: 182.182
[6000]  valid_set's l1: 180.598
```

```
[7000]  valid_set's l1: 179.259
[8000]  valid_set's l1: 177.947
[9000]  valid_set's l1: 177.153
[10000] valid_set's l1: 176.464
[1000]  valid_set's l1: 191.908
[2000]  valid_set's l1: 185.718
[3000]  valid_set's l1: 181.695
[4000]  valid_set's l1: 178.82
[5000]  valid_set's l1: 176.571
[6000]  valid_set's l1: 175.066
[7000]  valid_set's l1: 173.819
[8000]  valid_set's l1: 172.729
[9000]  valid_set's l1: 171.811
[10000] valid_set's l1: 170.954
[1000]  valid_set's l1: 199.123
[2000]  valid_set's l1: 192.568
[3000]  valid_set's l1: 187.619
[4000]  valid_set's l1: 184.365
[5000]  valid_set's l1: 181.748
[6000]  valid_set's l1: 179.978
[7000]  valid_set's l1: 178.273
[8000]  valid_set's l1: 176.707
[9000]  valid_set's l1: 175.611
[10000] valid_set's l1: 174.571
[1000]  valid_set's l1: 186.46
[2000]  valid_set's l1: 179.953
[3000]  valid_set's l1: 176.003
[4000]  valid_set's l1: 173.049
[5000]  valid_set's l1: 170.879
[6000]  valid_set's l1: 169.172
[7000]  valid_set's l1: 167.742
[8000]  valid_set's l1: 166.515
[9000]  valid_set's l1: 165.512
[10000] valid_set's l1: 164.636
[1000]  valid_set's l1: 191.178
[2000]  valid_set's l1: 184.287
[3000]  valid_set's l1: 180.194
[4000]  valid_set's l1: 177.31
[5000]  valid_set's l1: 175.582
[6000]  valid_set's l1: 174.015
[7000]  valid_set's l1: 172.688
[8000]  valid_set's l1: 171.419
[9000]  valid_set's l1: 170.527
[10000] valid_set's l1: 169.678
        -84.3514         = Validation score   (-mean_absolute_error)
        968.94s  = Training   runtime
        7.98s    = Validation runtime
```

```
Fitting model: WeightedEnsemble_L2 …
        -81.7967         = Validation score   (-mean_absolute_error)
        0.04s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 3383.58s … Best model:
"WeightedEnsemble_L2"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_133_A_seed_2/")
```

[27]:
```
loc = "B"
predictors[1] = fit_predictor_for_location(loc)
```

```
Warning: path already exists! This predictor may overwrite an existing
predictor! path="AutogluonModels/submission_133_B_seed_0"
Beginning AutoGluon training …
AutoGluon will save models to "AutogluonModels/submission_133_B_seed_0/"
AutoGluon Version:  0.8.2
Python Version:     3.10.12
Operating System:   Darwin
Platform Machine:   arm64
Platform Version:   Darwin Kernel Version 22.1.0: Sun Oct  9 20:15:09 PDT 2022;
root:xnu-8792.41.9~2/RELEASE_ARM64_T6000
Disk Space Avail:   110.62 GB / 494.38 GB (22.4%)
Train Data Rows:    27726
Train Data Columns: 44
Tuning Data Rows:    1500
Tuning Data Columns: 44
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and many unique label-values observed).
        Label info (max, min, mean, stddev): (1152.3, -0.0, 97.18811, 205.46465)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    3838.22 MB
        Train Data (Original)  Memory Usage: 11.75 MB (0.3% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 2 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
```

```
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 2): ['elevation:m', 'location']
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):

Training model for location B, seed 0…

                ('float', []) : 41 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])   :  1 | ['is_estimated']
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 40 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', ['bool']) :  2 | ['snow_density:kgm3', 'is_estimated']
        0.1s = Fit runtime
        42 features in original data used to generate 42 features in processed
data.
        Train Data (Processed) Memory Usage: 9.41 MB (0.2% of available memory)
Data preprocessing and feature engineering runtime = 0.14s …
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
use_bag_holdout=True, will use tuning_data as holdout (will not be used for
early stopping).
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'XT': [{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}},
{'extra_trees': True, 'feature_fraction': 0.7832570544199176, 'learning_rate':
0.021720607471727896, 'min_data_in_leaf': 3, 'num_leaves': 21, 'ag_args':
{'name_suffix': '_r118', 'priority': 17}}],
        'FASTAI': {},
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
```

```
Excluded models: ['KNN'] (Specified by `excluded_model_types`)
Fitting 5 L1 models …
Fitting model: LightGBMXT_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
[1000]  valid_set's l1: 24.9823
[2000]  valid_set's l1: 24.0733
[3000]  valid_set's l1: 23.6188
[4000]  valid_set's l1: 23.3861
[5000]  valid_set's l1: 23.2578
[6000]  valid_set's l1: 23.1168
[7000]  valid_set's l1: 23.038
[8000]  valid_set's l1: 22.9847
[9000]  valid_set's l1: 22.9359
[10000] valid_set's l1: 22.9005
[1000]  valid_set's l1: 25.9764
[2000]  valid_set's l1: 24.914
[3000]  valid_set's l1: 24.3027
[4000]  valid_set's l1: 23.9433
[5000]  valid_set's l1: 23.6719
[6000]  valid_set's l1: 23.5361
[7000]  valid_set's l1: 23.4171
[8000]  valid_set's l1: 23.3182
[9000]  valid_set's l1: 23.2334
[10000] valid_set's l1: 23.1685
[1000]  valid_set's l1: 25.1397
[2000]  valid_set's l1: 24.1607
[3000]  valid_set's l1: 23.6784
[4000]  valid_set's l1: 23.4348
[5000]  valid_set's l1: 23.2594
[6000]  valid_set's l1: 23.1322
[7000]  valid_set's l1: 23.064
[8000]  valid_set's l1: 23.0045
[9000]  valid_set's l1: 22.9752
[10000] valid_set's l1: 22.9468
[1000]  valid_set's l1: 25.3288
[2000]  valid_set's l1: 24.2643
[3000]  valid_set's l1: 23.7336
[4000]  valid_set's l1: 23.4957
[5000]  valid_set's l1: 23.3304
[6000]  valid_set's l1: 23.2166
[7000]  valid_set's l1: 23.1341
[8000]  valid_set's l1: 23.0805
[9000]  valid_set's l1: 23.0368
[10000] valid_set's l1: 23.0131
[1000]  valid_set's l1: 26.3129
[2000]  valid_set's l1: 25.1995
```

```
[3000]  valid_set's l1: 24.6592
[4000]  valid_set's l1: 24.3339
[5000]  valid_set's l1: 24.1686
[6000]  valid_set's l1: 24.041
[7000]  valid_set's l1: 23.9404
[8000]  valid_set's l1: 23.8814
[9000]  valid_set's l1: 23.8332
[10000] valid_set's l1: 23.7835
[1000]  valid_set's l1: 24.2321
[2000]  valid_set's l1: 23.4665
[3000]  valid_set's l1: 23.0633
[4000]  valid_set's l1: 22.8109
[5000]  valid_set's l1: 22.6737
[6000]  valid_set's l1: 22.5768
[7000]  valid_set's l1: 22.4997
[8000]  valid_set's l1: 22.4225
[9000]  valid_set's l1: 22.3669
[10000] valid_set's l1: 22.3357
[1000]  valid_set's l1: 26.5645
[2000]  valid_set's l1: 25.6793
[3000]  valid_set's l1: 25.2191
[4000]  valid_set's l1: 24.9509
[5000]  valid_set's l1: 24.7913
[6000]  valid_set's l1: 24.7034
[7000]  valid_set's l1: 24.6078
[8000]  valid_set's l1: 24.5502
[9000]  valid_set's l1: 24.4964
[10000] valid_set's l1: 24.4488
[1000]  valid_set's l1: 26.4109
[2000]  valid_set's l1: 25.3746
[3000]  valid_set's l1: 24.8301
[4000]  valid_set's l1: 24.5654
[5000]  valid_set's l1: 24.4091
[6000]  valid_set's l1: 24.2854
[7000]  valid_set's l1: 24.1809
[8000]  valid_set's l1: 24.0869
[9000]  valid_set's l1: 24.0485
[10000] valid_set's l1: 23.997
[1000]  valid_set's l1: 24.9739
[2000]  valid_set's l1: 24.0303
[3000]  valid_set's l1: 23.6008
[4000]  valid_set's l1: 23.3812
[5000]  valid_set's l1: 23.2164
[6000]  valid_set's l1: 23.1103
[7000]  valid_set's l1: 23.0547
[8000]  valid_set's l1: 22.9991
[9000]  valid_set's l1: 22.9657
[10000] valid_set's l1: 22.9467
```

```
[1000]  valid_set's l1: 25.465
[2000]  valid_set's l1: 24.6073
[3000]  valid_set's l1: 24.2125
[4000]  valid_set's l1: 23.9512
[5000]  valid_set's l1: 23.793
[6000]  valid_set's l1: 23.6824
[7000]  valid_set's l1: 23.5905
[8000]  valid_set's l1: 23.5351
[9000]  valid_set's l1: 23.4967
[10000] valid_set's l1: 23.4694
[1000]  valid_set's l1: 24.6048
[2000]  valid_set's l1: 23.4806
[3000]  valid_set's l1: 22.9534
[4000]  valid_set's l1: 22.6008
[5000]  valid_set's l1: 22.4251
[6000]  valid_set's l1: 22.258
[7000]  valid_set's l1: 22.1625
[8000]  valid_set's l1: 22.0964
[9000]  valid_set's l1: 22.0322
[10000] valid_set's l1: 21.9882
[1000]  valid_set's l1: 24.5681
[2000]  valid_set's l1: 23.7116
[3000]  valid_set's l1: 23.1753
[4000]  valid_set's l1: 22.913
[5000]  valid_set's l1: 22.7172
[6000]  valid_set's l1: 22.5783
[7000]  valid_set's l1: 22.4563
[8000]  valid_set's l1: 22.375
[9000]  valid_set's l1: 22.321
[10000] valid_set's l1: 22.2747

        -12.6882         = Validation score   (-mean_absolute_error)
        1188.99s         = Training   runtime
        11.66s   = Validation runtime
Fitting model: ExtraTreesMSE_BAG_L1 …
        -16.1928         = Validation score   (-mean_absolute_error)
        4.21s    = Training   runtime
        0.55s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -14.135  = Validation score   (-mean_absolute_error)
        361.34s  = Training   runtime
        0.83s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -12.7262         = Validation score   (-mean_absolute_error)
```

```
        865.19s  = Training   runtime
        0.43s    = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
[1000]  valid_set's l1: 26.5807
[2000]  valid_set's l1: 25.3328
[3000]  valid_set's l1: 24.6564
[4000]  valid_set's l1: 24.1772
[5000]  valid_set's l1: 23.848
[6000]  valid_set's l1: 23.6133
[7000]  valid_set's l1: 23.4074
[8000]  valid_set's l1: 23.2355
[9000]  valid_set's l1: 23.1023
[10000] valid_set's l1: 22.9847
[1000]  valid_set's l1: 27.9242
[2000]  valid_set's l1: 26.631
[3000]  valid_set's l1: 25.8685
[4000]  valid_set's l1: 25.2806
[5000]  valid_set's l1: 24.8644
[6000]  valid_set's l1: 24.5258
[7000]  valid_set's l1: 24.2422
[8000]  valid_set's l1: 24.0398
[9000]  valid_set's l1: 23.8779
[10000] valid_set's l1: 23.7357
[1000]  valid_set's l1: 26.9046
[2000]  valid_set's l1: 25.813
[3000]  valid_set's l1: 25.0918
[4000]  valid_set's l1: 24.5729
[5000]  valid_set's l1: 24.1882
[6000]  valid_set's l1: 23.9027
[7000]  valid_set's l1: 23.6903
[8000]  valid_set's l1: 23.5007
[9000]  valid_set's l1: 23.3663
[10000] valid_set's l1: 23.2498
[1000]  valid_set's l1: 27.2765
[2000]  valid_set's l1: 25.9482
[3000]  valid_set's l1: 25.1796
[4000]  valid_set's l1: 24.6232
[5000]  valid_set's l1: 24.2338
[6000]  valid_set's l1: 23.937
[7000]  valid_set's l1: 23.7382
[8000]  valid_set's l1: 23.5564
[9000]  valid_set's l1: 23.3992
[10000] valid_set's l1: 23.2767
[1000]  valid_set's l1: 27.7389
[2000]  valid_set's l1: 26.5469
```

```
[3000]  valid_set's l1: 25.8069
[4000]  valid_set's l1: 25.2497
[5000]  valid_set's l1: 24.869
[6000]  valid_set's l1: 24.557
[7000]  valid_set's l1: 24.307
[8000]  valid_set's l1: 24.1449
[9000]  valid_set's l1: 24
[10000] valid_set's l1: 23.8642
[1000]  valid_set's l1: 25.5749
[2000]  valid_set's l1: 24.4198
[3000]  valid_set's l1: 23.7406
[4000]  valid_set's l1: 23.2866
[5000]  valid_set's l1: 22.9634
[6000]  valid_set's l1: 22.7113
[7000]  valid_set's l1: 22.5061
[8000]  valid_set's l1: 22.3266
[9000]  valid_set's l1: 22.1598
[10000] valid_set's l1: 22.0314
[1000]  valid_set's l1: 27.7964
[2000]  valid_set's l1: 26.7183
[3000]  valid_set's l1: 26.1617
[4000]  valid_set's l1: 25.6769
[5000]  valid_set's l1: 25.3568
[6000]  valid_set's l1: 25.1163
[7000]  valid_set's l1: 24.92
[8000]  valid_set's l1: 24.7271
[9000]  valid_set's l1: 24.5843
[10000] valid_set's l1: 24.4382
[1000]  valid_set's l1: 27.9846
[2000]  valid_set's l1: 26.8726
[3000]  valid_set's l1: 26.0866
[4000]  valid_set's l1: 25.5619
[5000]  valid_set's l1: 25.1411
[6000]  valid_set's l1: 24.8243
[7000]  valid_set's l1: 24.553
[8000]  valid_set's l1: 24.3643
[9000]  valid_set's l1: 24.1824
[10000] valid_set's l1: 24.045
[1000]  valid_set's l1: 26.2792
[2000]  valid_set's l1: 25.1296
[3000]  valid_set's l1: 24.4846
[4000]  valid_set's l1: 23.9873
[5000]  valid_set's l1: 23.6254
[6000]  valid_set's l1: 23.3153
[7000]  valid_set's l1: 23.0801
[8000]  valid_set's l1: 22.9223
[9000]  valid_set's l1: 22.7924
[10000] valid_set's l1: 22.6706
```

```
[1000]  valid_set's l1: 27.2297
[2000]  valid_set's l1: 26.0516
[3000]  valid_set's l1: 25.3436
[4000]  valid_set's l1: 24.8427
[5000]  valid_set's l1: 24.4509
[6000]  valid_set's l1: 24.1611
[7000]  valid_set's l1: 23.9531
[8000]  valid_set's l1: 23.7518
[9000]  valid_set's l1: 23.6218
[10000] valid_set's l1: 23.4988
[1000]  valid_set's l1: 26.3919
[2000]  valid_set's l1: 25.1573
[3000]  valid_set's l1: 24.4281
[4000]  valid_set's l1: 23.8498
[5000]  valid_set's l1: 23.4129
[6000]  valid_set's l1: 23.0476
[7000]  valid_set's l1: 22.7823
[8000]  valid_set's l1: 22.589
[9000]  valid_set's l1: 22.4327
[10000] valid_set's l1: 22.3057
[1000]  valid_set's l1: 26.4825
[2000]  valid_set's l1: 25.2395
[3000]  valid_set's l1: 24.4583
[4000]  valid_set's l1: 23.9084
[5000]  valid_set's l1: 23.5302
[6000]  valid_set's l1: 23.234
[7000]  valid_set's l1: 23.0343
[8000]  valid_set's l1: 22.8474
[9000]  valid_set's l1: 22.6763
[10000] valid_set's l1: 22.5423

        -12.4199        = Validation score   (-mean_absolute_error)
        970.95s  = Training   runtime
        7.98s    = Validation runtime
Fitting model: WeightedEnsemble_L2 …
        -11.8414        = Validation score   (-mean_absolute_error)
        0.05s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 3431.83s … Best model:
"WeightedEnsemble_L2"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_133_B_seed_0/")
Beginning AutoGluon training …
AutoGluon will save models to "AutogluonModels/submission_133_B_seed_1/"
AutoGluon Version:  0.8.2
Python Version:     3.10.12
Operating System:   Darwin
Platform Machine:   arm64
```

```
Platform Version:   Darwin Kernel Version 22.1.0: Sun Oct  9 20:15:09 PDT 2022;
root:xnu-8792.41.9~2/RELEASE_ARM64_T6000
Disk Space Avail:   109.52 GB / 494.38 GB (22.2%)
Train Data Rows:    27726
Train Data Columns: 44
Tuning Data Rows:    1500
Tuning Data Columns: 44
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and many unique label-values observed).
        Label info (max, min, mean, stddev): (1152.3, -0.0, 97.2757, 205.47356)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    3993.34 MB
        Train Data (Original)  Memory Usage: 11.75 MB (0.3% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 2 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 2): ['elevation:m', 'location']
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 41 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])   :  1 | ['is_estimated']
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 40 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
```

```
                ('int', ['bool']) :  2 | ['snow_density:kgm3', 'is_estimated']
        0.1s = Fit runtime
        42 features in original data used to generate 42 features in processed
data.
        Train Data (Processed) Memory Usage: 9.41 MB (0.2% of available memory)
Data preprocessing and feature engineering runtime = 0.13s …
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
use_bag_holdout=True, will use tuning_data as holdout (will not be used for
early stopping).
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'XT': [{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}},
{'extra_trees': True, 'feature_fraction': 0.7832570544199176, 'learning_rate':
0.021720607471727896, 'min_data_in_leaf': 3, 'num_leaves': 21, 'ag_args':
{'name_suffix': '_r118', 'priority': 17}}],
        'FASTAI': {},
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
Excluded models: ['KNN'] (Specified by `excluded_model_types`)
Fitting 5 L1 models …
Fitting model: LightGBMXT_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy

Training model for location B, seed 1…
[1000]  valid_set's l1: 24.6747
[2000]  valid_set's l1: 23.8668
[3000]  valid_set's l1: 23.4528
[4000]  valid_set's l1: 23.235
[5000]  valid_set's l1: 23.0878
[6000]  valid_set's l1: 23.0001
[7000]  valid_set's l1: 22.9431
[8000]  valid_set's l1: 22.8927
[9000]  valid_set's l1: 22.8463
[10000] valid_set's l1: 22.8083
[1000]  valid_set's l1: 25.749
[2000]  valid_set's l1: 24.4924
[3000]  valid_set's l1: 23.9607
[4000]  valid_set's l1: 23.6463
[5000]  valid_set's l1: 23.3896
```

```
[6000]  valid_set's l1: 23.1807
[7000]  valid_set's l1: 23.0407
[8000]  valid_set's l1: 22.9308
[9000]  valid_set's l1: 22.8697
[10000] valid_set's l1: 22.8104
[1000]  valid_set's l1: 25.2678
[2000]  valid_set's l1: 24.3184
[3000]  valid_set's l1: 23.8835
[4000]  valid_set's l1: 23.6879
[5000]  valid_set's l1: 23.5162
[6000]  valid_set's l1: 23.4167
[7000]  valid_set's l1: 23.3194
[8000]  valid_set's l1: 23.2547
[9000]  valid_set's l1: 23.199
[10000] valid_set's l1: 23.1637
[1000]  valid_set's l1: 25.6554
[2000]  valid_set's l1: 24.5044
[3000]  valid_set's l1: 24.0448
[4000]  valid_set's l1: 23.7516
[5000]  valid_set's l1: 23.5735
[6000]  valid_set's l1: 23.4353
[7000]  valid_set's l1: 23.3754
[8000]  valid_set's l1: 23.3081
[9000]  valid_set's l1: 23.259
[10000] valid_set's l1: 23.2067
[1000]  valid_set's l1: 26.3226
[2000]  valid_set's l1: 25.2459
[3000]  valid_set's l1: 24.6854
[4000]  valid_set's l1: 24.3958
[5000]  valid_set's l1: 24.2344
[6000]  valid_set's l1: 24.0793
[7000]  valid_set's l1: 24.0118
[8000]  valid_set's l1: 23.9459
[9000]  valid_set's l1: 23.9002
[10000] valid_set's l1: 23.864
[1000]  valid_set's l1: 24.5377
[2000]  valid_set's l1: 23.694
[3000]  valid_set's l1: 23.3508
[4000]  valid_set's l1: 23.0765
[5000]  valid_set's l1: 22.9479
[6000]  valid_set's l1: 22.8148
[7000]  valid_set's l1: 22.7228
[8000]  valid_set's l1: 22.6622
[9000]  valid_set's l1: 22.6233
[10000] valid_set's l1: 22.5632
[1000]  valid_set's l1: 26.7913
[2000]  valid_set's l1: 25.8794
[3000]  valid_set's l1: 25.3998
```

```
[4000]  valid_set's l1: 25.1423
[5000]  valid_set's l1: 24.9777
[6000]  valid_set's l1: 24.8506
[7000]  valid_set's l1: 24.7417
[8000]  valid_set's l1: 24.6708
[9000]  valid_set's l1: 24.6051
[10000] valid_set's l1: 24.5582
[1000]  valid_set's l1: 25.9854
[2000]  valid_set's l1: 25.0042
[3000]  valid_set's l1: 24.4937
[4000]  valid_set's l1: 24.2062
[5000]  valid_set's l1: 23.9621
[6000]  valid_set's l1: 23.8488
[7000]  valid_set's l1: 23.7913
[8000]  valid_set's l1: 23.7272
[9000]  valid_set's l1: 23.6798
[10000] valid_set's l1: 23.6431
[1000]  valid_set's l1: 24.7444
[2000]  valid_set's l1: 23.8759
[3000]  valid_set's l1: 23.4811
[4000]  valid_set's l1: 23.194
[5000]  valid_set's l1: 23.021
[6000]  valid_set's l1: 22.8978
[7000]  valid_set's l1: 22.7991
[8000]  valid_set's l1: 22.7463
[9000]  valid_set's l1: 22.6987
[10000] valid_set's l1: 22.6702
[1000]  valid_set's l1: 25.4104
[2000]  valid_set's l1: 24.469
[3000]  valid_set's l1: 23.919
[4000]  valid_set's l1: 23.6621
[5000]  valid_set's l1: 23.49
[6000]  valid_set's l1: 23.3699
[7000]  valid_set's l1: 23.2711
[8000]  valid_set's l1: 23.2098
[9000]  valid_set's l1: 23.1741
[10000] valid_set's l1: 23.1318
[1000]  valid_set's l1: 24.7188
[2000]  valid_set's l1: 23.6166
[3000]  valid_set's l1: 23.0316
[4000]  valid_set's l1: 22.6635
[5000]  valid_set's l1: 22.4135
[6000]  valid_set's l1: 22.255
[7000]  valid_set's l1: 22.1592
[8000]  valid_set's l1: 22.0711
[9000]  valid_set's l1: 22.0182
[10000] valid_set's l1: 21.9622
[1000]  valid_set's l1: 25.3163
```

```
[2000]  valid_set's l1: 24.3057
[3000]  valid_set's l1: 23.8426
[4000]  valid_set's l1: 23.6067
[5000]  valid_set's l1: 23.3966
[6000]  valid_set's l1: 23.2264
[7000]  valid_set's l1: 23.1155
[8000]  valid_set's l1: 23.0366
[9000]  valid_set's l1: 22.9821
[10000] valid_set's l1: 22.9293
        -13.2784         = Validation score   (-mean_absolute_error)
        1130.7s  = Training   runtime
        12.04s   = Validation runtime
Fitting model: ExtraTreesMSE_BAG_L1 …
        -15.7629         = Validation score   (-mean_absolute_error)
        3.8s     = Training   runtime
        0.51s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -14.0761         = Validation score   (-mean_absolute_error)
        379.33s  = Training   runtime
        0.86s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -12.8943         = Validation score   (-mean_absolute_error)
        896.53s  = Training   runtime
        0.51s    = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
[1000]  valid_set's l1: 26.4789
[2000]  valid_set's l1: 25.2019
[3000]  valid_set's l1: 24.474
[4000]  valid_set's l1: 23.9805
[5000]  valid_set's l1: 23.638
[6000]  valid_set's l1: 23.3332
[7000]  valid_set's l1: 23.1194
[8000]  valid_set's l1: 22.9506
[9000]  valid_set's l1: 22.8094
[10000] valid_set's l1: 22.6868
[1000]  valid_set's l1: 27.8415
[2000]  valid_set's l1: 26.3876
[3000]  valid_set's l1: 25.6071
[4000]  valid_set's l1: 25.0144
[5000]  valid_set's l1: 24.5754
[6000]  valid_set's l1: 24.2373
```

```
[7000]  valid_set's l1: 23.9874
[8000]  valid_set's l1: 23.749
[9000]  valid_set's l1: 23.5657
[10000] valid_set's l1: 23.4136
[1000]  valid_set's l1: 26.7741
[2000]  valid_set's l1: 25.6359
[3000]  valid_set's l1: 25.0144
[4000]  valid_set's l1: 24.5765
[5000]  valid_set's l1: 24.224
[6000]  valid_set's l1: 23.9409
[7000]  valid_set's l1: 23.6867
[8000]  valid_set's l1: 23.4965
[9000]  valid_set's l1: 23.3472
[10000] valid_set's l1: 23.2383
[1000]  valid_set's l1: 27.1439
[2000]  valid_set's l1: 25.9572
[3000]  valid_set's l1: 25.1002
[4000]  valid_set's l1: 24.5381
[5000]  valid_set's l1: 24.1316
[6000]  valid_set's l1: 23.8517
[7000]  valid_set's l1: 23.6175
[8000]  valid_set's l1: 23.4464
[9000]  valid_set's l1: 23.2976
[10000] valid_set's l1: 23.1887
[1000]  valid_set's l1: 27.6514
[2000]  valid_set's l1: 26.4474
[3000]  valid_set's l1: 25.7453
[4000]  valid_set's l1: 25.1609
[5000]  valid_set's l1: 24.7215
[6000]  valid_set's l1: 24.4502
[7000]  valid_set's l1: 24.231
[8000]  valid_set's l1: 24.0386
[9000]  valid_set's l1: 23.889
[10000] valid_set's l1: 23.7628
[1000]  valid_set's l1: 25.9126
[2000]  valid_set's l1: 24.7602
[3000]  valid_set's l1: 24.0768
[4000]  valid_set's l1: 23.5667
[5000]  valid_set's l1: 23.2258
[6000]  valid_set's l1: 22.9642
[7000]  valid_set's l1: 22.7572
[8000]  valid_set's l1: 22.5686
[9000]  valid_set's l1: 22.441
[10000] valid_set's l1: 22.3423
[1000]  valid_set's l1: 28.1131
[2000]  valid_set's l1: 27.0096
[3000]  valid_set's l1: 26.4407
[4000]  valid_set's l1: 25.9647
```

```
[5000]  valid_set's l1: 25.6824
[6000]  valid_set's l1: 25.4373
[7000]  valid_set's l1: 25.2596
[8000]  valid_set's l1: 25.0594
[9000]  valid_set's l1: 24.9241
[10000] valid_set's l1: 24.8193
[1000]  valid_set's l1: 27.8502
[2000]  valid_set's l1: 26.6121
[3000]  valid_set's l1: 25.8256
[4000]  valid_set's l1: 25.3777
[5000]  valid_set's l1: 24.9755
[6000]  valid_set's l1: 24.6828
[7000]  valid_set's l1: 24.4664
[8000]  valid_set's l1: 24.2457
[9000]  valid_set's l1: 24.1022
[10000] valid_set's l1: 23.9581
[1000]  valid_set's l1: 25.9728
[2000]  valid_set's l1: 24.8519
[3000]  valid_set's l1: 24.1959
[4000]  valid_set's l1: 23.7272
[5000]  valid_set's l1: 23.3665
[6000]  valid_set's l1: 23.0863
[7000]  valid_set's l1: 22.8708
[8000]  valid_set's l1: 22.6829
[9000]  valid_set's l1: 22.5368
[10000] valid_set's l1: 22.4084
[1000]  valid_set's l1: 27.0978
[2000]  valid_set's l1: 25.8635
[3000]  valid_set's l1: 25.0909
[4000]  valid_set's l1: 24.5866
[5000]  valid_set's l1: 24.191
[6000]  valid_set's l1: 23.8919
[7000]  valid_set's l1: 23.6804
[8000]  valid_set's l1: 23.4896
[9000]  valid_set's l1: 23.3443
[10000] valid_set's l1: 23.2171
[1000]  valid_set's l1: 26.4579
[2000]  valid_set's l1: 25.1641
[3000]  valid_set's l1: 24.3639
[4000]  valid_set's l1: 23.8413
[5000]  valid_set's l1: 23.459
[6000]  valid_set's l1: 23.1096
[7000]  valid_set's l1: 22.8566
[8000]  valid_set's l1: 22.6422
[9000]  valid_set's l1: 22.4852
[10000] valid_set's l1: 22.3476
[1000]  valid_set's l1: 26.6843
[2000]  valid_set's l1: 25.3016
```

```
[3000]  valid_set's l1: 24.5302
[4000]  valid_set's l1: 23.9536
[5000]  valid_set's l1: 23.5614
[6000]  valid_set's l1: 23.2537
[7000]  valid_set's l1: 23.0121
[8000]  valid_set's l1: 22.8257
[9000]  valid_set's l1: 22.6619
[10000] valid_set's l1: 22.5121
        -13.1122         = Validation score   (-mean_absolute_error)
        899.74s  = Training   runtime
        7.66s    = Validation runtime
Fitting model: WeightedEnsemble_L2 …
        -12.4632         = Validation score   (-mean_absolute_error)
        0.14s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 3351.64s … Best model:
"WeightedEnsemble_L2"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_133_B_seed_1/")
Beginning AutoGluon training …
AutoGluon will save models to "AutogluonModels/submission_133_B_seed_2/"
AutoGluon Version:  0.8.2
Python Version:     3.10.12
Operating System:   Darwin
Platform Machine:   arm64
Platform Version:   Darwin Kernel Version 22.1.0: Sun Oct  9 20:15:09 PDT 2022;
root:xnu-8792.41.9~2/RELEASE_ARM64_T6000
Disk Space Avail:   108.60 GB / 494.38 GB (22.0%)
Train Data Rows:    27726
Train Data Columns: 44
Tuning Data Rows:    1500
Tuning Data Columns: 44
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and many unique label-values observed).
        Label info (max, min, mean, stddev): (1152.3, -0.0, 97.48588, 205.77547)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    4343.3 MB
        Train Data (Original)  Memory Usage: 11.75 MB (0.3% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
```

```
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 2 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 2): ['elevation:m', 'location']
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 41 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])   :  1 | ['is_estimated']
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 40 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', ['bool']) :  2 | ['snow_density:kgm3', 'is_estimated']
        0.1s = Fit runtime
        42 features in original data used to generate 42 features in processed
data.
        Train Data (Processed) Memory Usage: 9.41 MB (0.2% of available memory)
Data preprocessing and feature engineering runtime = 0.17s …
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()

Training model for location B, seed 2…

use_bag_holdout=True, will use tuning_data as holdout (will not be used for
early stopping).
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'XT': [{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}},
{'extra_trees': True, 'feature_fraction': 0.7832570544199176, 'learning_rate':
```

```
0.021720607471727896, 'min_data_in_leaf': 3, 'num_leaves': 21, 'ag_args':
{'name_suffix': '_r118', 'priority': 17}}],
        'FASTAI': {},
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
Excluded models: ['KNN'] (Specified by `excluded_model_types`)
Fitting 5 L1 models …
Fitting model: LightGBMXT_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy

[1000]  valid_set's l1: 24.8817
[2000]  valid_set's l1: 24.0041
[3000]  valid_set's l1: 23.4557
[4000]  valid_set's l1: 23.1546
[5000]  valid_set's l1: 22.9885
[6000]  valid_set's l1: 22.8626
[7000]  valid_set's l1: 22.8011
[8000]  valid_set's l1: 22.7533
[9000]  valid_set's l1: 22.7102
[10000] valid_set's l1: 22.6793
[1000]  valid_set's l1: 25.632
[2000]  valid_set's l1: 24.5094
[3000]  valid_set's l1: 23.9587
[4000]  valid_set's l1: 23.637
[5000]  valid_set's l1: 23.3853
[6000]  valid_set's l1: 23.2465
[7000]  valid_set's l1: 23.1427
[8000]  valid_set's l1: 23.0697
[9000]  valid_set's l1: 23.0084
[10000] valid_set's l1: 22.9537
[1000]  valid_set's l1: 25.4171
[2000]  valid_set's l1: 24.4698
[3000]  valid_set's l1: 23.9981
[4000]  valid_set's l1: 23.7147
[5000]  valid_set's l1: 23.5678
[6000]  valid_set's l1: 23.4733
[7000]  valid_set's l1: 23.3967
[8000]  valid_set's l1: 23.3379
[9000]  valid_set's l1: 23.2925
[10000] valid_set's l1: 23.2537
[1000]  valid_set's l1: 25.6286
[2000]  valid_set's l1: 24.6894
[3000]  valid_set's l1: 24.3332
[4000]  valid_set's l1: 24.0428
[5000]  valid_set's l1: 23.8833
[6000]  valid_set's l1: 23.7404
```

```
[7000]  valid_set's l1: 23.6553
[8000]  valid_set's l1: 23.5829
[9000]  valid_set's l1: 23.5358
[10000] valid_set's l1: 23.4995
[1000]  valid_set's l1: 26.0677
[2000]  valid_set's l1: 24.9848
[3000]  valid_set's l1: 24.5486
[4000]  valid_set's l1: 24.2619
[5000]  valid_set's l1: 24.0809
[6000]  valid_set's l1: 23.9754
[7000]  valid_set's l1: 23.8796
[8000]  valid_set's l1: 23.8036
[9000]  valid_set's l1: 23.7575
[10000] valid_set's l1: 23.7119
[1000]  valid_set's l1: 24.4347
[2000]  valid_set's l1: 23.5234
[3000]  valid_set's l1: 23.0597
[4000]  valid_set's l1: 22.7847
[5000]  valid_set's l1: 22.6349
[6000]  valid_set's l1: 22.5438
[7000]  valid_set's l1: 22.4453
[8000]  valid_set's l1: 22.3543
[9000]  valid_set's l1: 22.3052
[10000] valid_set's l1: 22.27
[1000]  valid_set's l1: 26.5438
[2000]  valid_set's l1: 25.7321
[3000]  valid_set's l1: 25.3097
[4000]  valid_set's l1: 25.0697
[5000]  valid_set's l1: 24.8986
[6000]  valid_set's l1: 24.771
[7000]  valid_set's l1: 24.6828
[8000]  valid_set's l1: 24.6191
[9000]  valid_set's l1: 24.5723
[10000] valid_set's l1: 24.5288
[1000]  valid_set's l1: 25.9224
[2000]  valid_set's l1: 24.8084
[3000]  valid_set's l1: 24.356
[4000]  valid_set's l1: 24.0648
[5000]  valid_set's l1: 23.9081
[6000]  valid_set's l1: 23.7511
[7000]  valid_set's l1: 23.6431
[8000]  valid_set's l1: 23.5589
[9000]  valid_set's l1: 23.4994
[10000] valid_set's l1: 23.4657
[1000]  valid_set's l1: 24.7747
[2000]  valid_set's l1: 23.9975
[3000]  valid_set's l1: 23.605
[4000]  valid_set's l1: 23.3293
```

```
[5000]  valid_set's l1: 23.19
[6000]  valid_set's l1: 23.0596
[7000]  valid_set's l1: 22.9621
[8000]  valid_set's l1: 22.8936
[9000]  valid_set's l1: 22.8706
[10000] valid_set's l1: 22.8305
[1000]  valid_set's l1: 25.392
[2000]  valid_set's l1: 24.4399
[3000]  valid_set's l1: 23.9386
[4000]  valid_set's l1: 23.6806
[5000]  valid_set's l1: 23.5029
[6000]  valid_set's l1: 23.3838
[7000]  valid_set's l1: 23.3069
[8000]  valid_set's l1: 23.2263
[9000]  valid_set's l1: 23.1573
[10000] valid_set's l1: 23.1141
[1000]  valid_set's l1: 24.9217
[2000]  valid_set's l1: 23.8374
[3000]  valid_set's l1: 23.2302
[4000]  valid_set's l1: 22.8755
[5000]  valid_set's l1: 22.6671
[6000]  valid_set's l1: 22.5352
[7000]  valid_set's l1: 22.4222
[8000]  valid_set's l1: 22.319
[9000]  valid_set's l1: 22.276
[10000] valid_set's l1: 22.2125
[1000]  valid_set's l1: 24.8328
[2000]  valid_set's l1: 23.8752
[3000]  valid_set's l1: 23.4394
[4000]  valid_set's l1: 23.145
[5000]  valid_set's l1: 22.9683
[6000]  valid_set's l1: 22.8267
[7000]  valid_set's l1: 22.7061
[8000]  valid_set's l1: 22.6373
[9000]  valid_set's l1: 22.5818
[10000] valid_set's l1: 22.5325
        -11.5982         = Validation score   (-mean_absolute_error)
        1212.5s  = Training   runtime
        11.26s   = Validation runtime
Fitting model: ExtraTreesMSE_BAG_L1 …
        -14.2436         = Validation score   (-mean_absolute_error)
        3.88s    = Training   runtime
        0.57s    = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -11.9235         = Validation score   (-mean_absolute_error)
```

```
        387.64s  = Training   runtime
        0.88s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -11.3654         = Validation score   (-mean_absolute_error)
        919.56s  = Training   runtime
        0.52s    = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy

[1000]  valid_set's l1: 26.7657
[2000]  valid_set's l1: 25.5339
[3000]  valid_set's l1: 24.815
[4000]  valid_set's l1: 24.2863
[5000]  valid_set's l1: 23.9276
[6000]  valid_set's l1: 23.6947
[7000]  valid_set's l1: 23.4542
[8000]  valid_set's l1: 23.284
[9000]  valid_set's l1: 23.1504
[10000] valid_set's l1: 23.0388
[1000]  valid_set's l1: 27.7899
[2000]  valid_set's l1: 26.3148
[3000]  valid_set's l1: 25.44
[4000]  valid_set's l1: 24.8632
[5000]  valid_set's l1: 24.4119
[6000]  valid_set's l1: 24.1033
[7000]  valid_set's l1: 23.8573
[8000]  valid_set's l1: 23.6573
[9000]  valid_set's l1: 23.4732
[10000] valid_set's l1: 23.3198
[1000]  valid_set's l1: 27.0932
[2000]  valid_set's l1: 25.9403
[3000]  valid_set's l1: 25.2245
[4000]  valid_set's l1: 24.7475
[5000]  valid_set's l1: 24.4144
[6000]  valid_set's l1: 24.1134
[7000]  valid_set's l1: 23.8761
[8000]  valid_set's l1: 23.6876
[9000]  valid_set's l1: 23.5422
[10000] valid_set's l1: 23.4157
[1000]  valid_set's l1: 27.4011
[2000]  valid_set's l1: 26.0944
[3000]  valid_set's l1: 25.2756
[4000]  valid_set's l1: 24.7473
[5000]  valid_set's l1: 24.3134
[6000]  valid_set's l1: 24.0219
```

```
[7000]  valid_set's l1: 23.772
[8000]  valid_set's l1: 23.5874
[9000]  valid_set's l1: 23.4481
[10000] valid_set's l1: 23.3337
[1000]  valid_set's l1: 27.6709
[2000]  valid_set's l1: 26.5159
[3000]  valid_set's l1: 25.7703
[4000]  valid_set's l1: 25.2065
[5000]  valid_set's l1: 24.7806
[6000]  valid_set's l1: 24.4552
[7000]  valid_set's l1: 24.1918
[8000]  valid_set's l1: 24.0044
[9000]  valid_set's l1: 23.8392
[10000] valid_set's l1: 23.7324
[1000]  valid_set's l1: 26.2294
[2000]  valid_set's l1: 25.0695
[3000]  valid_set's l1: 24.469
[4000]  valid_set's l1: 23.9883
[5000]  valid_set's l1: 23.6366
[6000]  valid_set's l1: 23.3978
[7000]  valid_set's l1: 23.1956
[8000]  valid_set's l1: 23.0189
[9000]  valid_set's l1: 22.8799
[10000] valid_set's l1: 22.7359
[1000]  valid_set's l1: 27.9637
[2000]  valid_set's l1: 26.9051
[3000]  valid_set's l1: 26.2461
[4000]  valid_set's l1: 25.7779
[5000]  valid_set's l1: 25.4266
[6000]  valid_set's l1: 25.1565
[7000]  valid_set's l1: 24.9591
[8000]  valid_set's l1: 24.7789
[9000]  valid_set's l1: 24.6305
[10000] valid_set's l1: 24.5163
[1000]  valid_set's l1: 28.0985
[2000]  valid_set's l1: 26.8301
[3000]  valid_set's l1: 26.1555
[4000]  valid_set's l1: 25.5888
[5000]  valid_set's l1: 25.2572
[6000]  valid_set's l1: 24.9348
[7000]  valid_set's l1: 24.7053
[8000]  valid_set's l1: 24.5094
[9000]  valid_set's l1: 24.3407
[10000] valid_set's l1: 24.1956
[1000]  valid_set's l1: 26.2973
[2000]  valid_set's l1: 25.1127
[3000]  valid_set's l1: 24.3642
[4000]  valid_set's l1: 23.8828
```

```
[5000]  valid_set's l1: 23.5353
[6000]  valid_set's l1: 23.2419
[7000]  valid_set's l1: 23.0452
[8000]  valid_set's l1: 22.8462
[9000]  valid_set's l1: 22.6937
[10000] valid_set's l1: 22.5748
[1000]  valid_set's l1: 27.2298
[2000]  valid_set's l1: 26.0007
[3000]  valid_set's l1: 25.2365
[4000]  valid_set's l1: 24.7577
[5000]  valid_set's l1: 24.3553
[6000]  valid_set's l1: 24.0534
[7000]  valid_set's l1: 23.8621
[8000]  valid_set's l1: 23.7059
[9000]  valid_set's l1: 23.5622
[10000] valid_set's l1: 23.4275
[1000]  valid_set's l1: 26.7974
[2000]  valid_set's l1: 25.4496
[3000]  valid_set's l1: 24.6268
[4000]  valid_set's l1: 24.0566
[5000]  valid_set's l1: 23.6781
[6000]  valid_set's l1: 23.3636
[7000]  valid_set's l1: 23.1543
[8000]  valid_set's l1: 22.9194
[9000]  valid_set's l1: 22.7755
[10000] valid_set's l1: 22.6183
[1000]  valid_set's l1: 26.7526
[2000]  valid_set's l1: 25.5802
[3000]  valid_set's l1: 24.7259
[4000]  valid_set's l1: 24.1721
[5000]  valid_set's l1: 23.7668
[6000]  valid_set's l1: 23.4411
[7000]  valid_set's l1: 23.1901
[8000]  valid_set's l1: 23.0023
[9000]  valid_set's l1: 22.8466
[10000] valid_set's l1: 22.7415
        -11.1633         = Validation score   (-mean_absolute_error)
        1042.54s         = Training   runtime
        7.35s    = Validation runtime
Fitting model: WeightedEnsemble_L2 …
        -10.6088         = Validation score   (-mean_absolute_error)
        0.05s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 3606.3s … Best model:
"WeightedEnsemble_L2"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_133_B_seed_2/")
```

[28]:
```
loc = "C"
predictors[2] = fit_predictor_for_location(loc)
```

```
Warning: path already exists! This predictor may overwrite an existing
predictor! path="AutogluonModels/submission_133_C_seed_0"
Beginning AutoGluon training …
AutoGluon will save models to "AutogluonModels/submission_133_C_seed_0/"
AutoGluon Version:  0.8.2
Python Version:     3.10.12
Operating System:   Darwin
Platform Machine:   arm64
Platform Version:   Darwin Kernel Version 22.1.0: Sun Oct  9 20:15:09 PDT 2022;
root:xnu-8792.41.9~2/RELEASE_ARM64_T6000
Disk Space Avail:   107.73 GB / 494.38 GB (21.8%)
Train Data Rows:    24416
Train Data Columns: 44
Tuning Data Rows:    1500
Tuning Data Columns: 44
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and label-values can't be converted to int).
        Label info (max, min, mean, stddev): (999.6, -0.0, 80.49013, 169.23435)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    4754.54 MB
        Train Data (Original)  Memory Usage: 10.42 MB (0.2% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 2 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 3): ['elevation:m', 'snow_drift:idx',
'location']
                These features carry no predictive signal and should be manually
investigated.
```

```
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 40 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])   :  1 | ['is_estimated']
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 39 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', ['bool']) :  2 | ['snow_density:kgm3', 'is_estimated']
        0.1s = Fit runtime
        41 features in original data used to generate 41 features in processed
data.

Training model for location C, seed 0…

        Train Data (Processed) Memory Usage: 8.14 MB (0.2% of available memory)
Data preprocessing and feature engineering runtime = 0.17s …
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
use_bag_holdout=True, will use tuning_data as holdout (will not be used for
early stopping).
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'XT': [{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}},
{'extra_trees': True, 'feature_fraction': 0.7832570544199176, 'learning_rate':
0.021720607471727896, 'min_data_in_leaf': 3, 'num_leaves': 21, 'ag_args':
{'name_suffix': '_r118', 'priority': 17}}],
        'FASTAI': {},
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
Excluded models: ['XT'] (Specified by `excluded_model_types`)
Fitting 6 L1 models …
Fitting model: KNeighborsUnif_BAG_L1 …
        -19.9796         = Validation score   (-mean_absolute_error)
        0.02s    = Training   runtime
        300.03s  = Validation runtime
Fitting model: KNeighborsDist_BAG_L1 …
        -20.0942         = Validation score   (-mean_absolute_error)
```

```
        0.02s    = Training   runtime
        297.08s  = Validation runtime
Fitting model: LightGBMXT_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
[1000]  valid_set's l1: 19.0604
[2000]  valid_set's l1: 18.3372
[3000]  valid_set's l1: 18.035
[4000]  valid_set's l1: 17.8967
[5000]  valid_set's l1: 17.8264
[6000]  valid_set's l1: 17.7994
[7000]  valid_set's l1: 17.7594
[8000]  valid_set's l1: 17.7282
[9000]  valid_set's l1: 17.7076
[10000] valid_set's l1: 17.6976
[1000]  valid_set's l1: 20.5522
[2000]  valid_set's l1: 19.9167
[3000]  valid_set's l1: 19.6157
[4000]  valid_set's l1: 19.3853
[5000]  valid_set's l1: 19.2678
[6000]  valid_set's l1: 19.189
[7000]  valid_set's l1: 19.148
[8000]  valid_set's l1: 19.0968
[9000]  valid_set's l1: 19.0644
[10000] valid_set's l1: 19.0423
[1000]  valid_set's l1: 19.1388
[2000]  valid_set's l1: 18.6351
[3000]  valid_set's l1: 18.4208
[4000]  valid_set's l1: 18.2987
[5000]  valid_set's l1: 18.2434
[6000]  valid_set's l1: 18.1908
[7000]  valid_set's l1: 18.1321
[8000]  valid_set's l1: 18.1059
[9000]  valid_set's l1: 18.0716
[10000] valid_set's l1: 18.055
[1000]  valid_set's l1: 19.762
[2000]  valid_set's l1: 19.149
[3000]  valid_set's l1: 18.855
[4000]  valid_set's l1: 18.6853
[5000]  valid_set's l1: 18.593
[6000]  valid_set's l1: 18.5192
[7000]  valid_set's l1: 18.5013
[8000]  valid_set's l1: 18.4701
[9000]  valid_set's l1: 18.4504
[10000] valid_set's l1: 18.4387
[1000]  valid_set's l1: 18.9891
[2000]  valid_set's l1: 18.4542
```

```
[3000]  valid_set's l1: 18.2179
[4000]  valid_set's l1: 18.0734
[5000]  valid_set's l1: 17.9198
[6000]  valid_set's l1: 17.8266
[7000]  valid_set's l1: 17.7834
[8000]  valid_set's l1: 17.7382
[9000]  valid_set's l1: 17.7168
[10000] valid_set's l1: 17.6957
[1000]  valid_set's l1: 19.9631
[2000]  valid_set's l1: 19.4
[3000]  valid_set's l1: 19.1431
[4000]  valid_set's l1: 19.02
[5000]  valid_set's l1: 18.9326
[6000]  valid_set's l1: 18.8414
[7000]  valid_set's l1: 18.7887
[8000]  valid_set's l1: 18.7528
[9000]  valid_set's l1: 18.7181
[10000] valid_set's l1: 18.6992
[1000]  valid_set's l1: 19.0887
[2000]  valid_set's l1: 18.4338
[3000]  valid_set's l1: 18.1387
[4000]  valid_set's l1: 17.9659
[5000]  valid_set's l1: 17.8461
[6000]  valid_set's l1: 17.7809
[7000]  valid_set's l1: 17.7324
[8000]  valid_set's l1: 17.6891
[9000]  valid_set's l1: 17.6653
[10000] valid_set's l1: 17.6441
[1000]  valid_set's l1: 19.6523
[2000]  valid_set's l1: 19.1642
[3000]  valid_set's l1: 18.9651
[4000]  valid_set's l1: 18.8194
[5000]  valid_set's l1: 18.7026
[6000]  valid_set's l1: 18.635
[7000]  valid_set's l1: 18.5785
[8000]  valid_set's l1: 18.5455
[9000]  valid_set's l1: 18.5086
[10000] valid_set's l1: 18.4894
[1000]  valid_set's l1: 19.6364
[2000]  valid_set's l1: 19.0075
[3000]  valid_set's l1: 18.7543
[4000]  valid_set's l1: 18.5788
[5000]  valid_set's l1: 18.4686
[6000]  valid_set's l1: 18.4382
[7000]  valid_set's l1: 18.3922
[8000]  valid_set's l1: 18.3557
[9000]  valid_set's l1: 18.3415
[10000] valid_set's l1: 18.3219
```

```
[1000]  valid_set's l1: 19.6527
[2000]  valid_set's l1: 18.9105
[3000]  valid_set's l1: 18.6017
[4000]  valid_set's l1: 18.4968
[5000]  valid_set's l1: 18.4176
[6000]  valid_set's l1: 18.3853
[7000]  valid_set's l1: 18.3548
[8000]  valid_set's l1: 18.3309
[9000]  valid_set's l1: 18.3217
[10000] valid_set's l1: 18.3043
[1000]  valid_set's l1: 20.2933
[2000]  valid_set's l1: 19.6015
[3000]  valid_set's l1: 19.338
[4000]  valid_set's l1: 19.1231
[5000]  valid_set's l1: 19.0268
[6000]  valid_set's l1: 18.9532
[7000]  valid_set's l1: 18.9124
[8000]  valid_set's l1: 18.8923
[9000]  valid_set's l1: 18.8688
[10000] valid_set's l1: 18.8491
[1000]  valid_set's l1: 19.4745
[2000]  valid_set's l1: 18.9264
[3000]  valid_set's l1: 18.6309
[4000]  valid_set's l1: 18.4313
[5000]  valid_set's l1: 18.3124
[6000]  valid_set's l1: 18.2409
[7000]  valid_set's l1: 18.1873
[8000]  valid_set's l1: 18.152
[9000]  valid_set's l1: 18.1258
[10000] valid_set's l1: 18.1097
        -11.6668         = Validation score   (-mean_absolute_error)
        1199.57s         = Training   runtime
        10.27s   = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -13.4241         = Validation score   (-mean_absolute_error)
        305.2s   = Training   runtime
        0.69s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -13.0767         = Validation score   (-mean_absolute_error)
        619.67s  = Training   runtime
        0.44s    = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
```

```
SequentialLocalFoldFittingStrategy
[1000]  valid_set's l1: 20.5519
[2000]  valid_set's l1: 19.6424
[3000]  valid_set's l1: 19.0885
[4000]  valid_set's l1: 18.6978
[5000]  valid_set's l1: 18.4319
[6000]  valid_set's l1: 18.2285
[7000]  valid_set's l1: 18.0792
[8000]  valid_set's l1: 17.9439
[9000]  valid_set's l1: 17.8392
[10000] valid_set's l1: 17.7447
[1000]  valid_set's l1: 21.1822
[2000]  valid_set's l1: 20.3379
[3000]  valid_set's l1: 19.7785
[4000]  valid_set's l1: 19.4081
[5000]  valid_set's l1: 19.1413
[6000]  valid_set's l1: 18.9608
[7000]  valid_set's l1: 18.8096
[8000]  valid_set's l1: 18.6767
[9000]  valid_set's l1: 18.5895
[10000] valid_set's l1: 18.5092
[1000]  valid_set's l1: 19.9299
[2000]  valid_set's l1: 19.1379
[3000]  valid_set's l1: 18.7247
[4000]  valid_set's l1: 18.4603
[5000]  valid_set's l1: 18.2636
[6000]  valid_set's l1: 18.1074
[7000]  valid_set's l1: 17.9822
[8000]  valid_set's l1: 17.8756
[9000]  valid_set's l1: 17.7827
[10000] valid_set's l1: 17.7054
[1000]  valid_set's l1: 20.5893
[2000]  valid_set's l1: 19.7714
[3000]  valid_set's l1: 19.3109
[4000]  valid_set's l1: 18.9869
[5000]  valid_set's l1: 18.7877
[6000]  valid_set's l1: 18.6242
[7000]  valid_set's l1: 18.4962
[8000]  valid_set's l1: 18.3948
[9000]  valid_set's l1: 18.3085
[10000] valid_set's l1: 18.2426
[1000]  valid_set's l1: 19.7704
[2000]  valid_set's l1: 19.1296
[3000]  valid_set's l1: 18.6951
[4000]  valid_set's l1: 18.4083
[5000]  valid_set's l1: 18.1855
[6000]  valid_set's l1: 18.019
```

```
[7000]  valid_set's l1: 17.9165
[8000]  valid_set's l1: 17.8067
[9000]  valid_set's l1: 17.7221
[10000] valid_set's l1: 17.6411
[1000]  valid_set's l1: 21.1664
[2000]  valid_set's l1: 20.4063
[3000]  valid_set's l1: 19.9303
[4000]  valid_set's l1: 19.6217
[5000]  valid_set's l1: 19.3808
[6000]  valid_set's l1: 19.2136
[7000]  valid_set's l1: 19.082
[8000]  valid_set's l1: 18.9817
[9000]  valid_set's l1: 18.9191
[10000] valid_set's l1: 18.8525
[1000]  valid_set's l1: 20.2505
[2000]  valid_set's l1: 19.3291
[3000]  valid_set's l1: 18.7909
[4000]  valid_set's l1: 18.4562
[5000]  valid_set's l1: 18.1675
[6000]  valid_set's l1: 18.019
[7000]  valid_set's l1: 17.8426
[8000]  valid_set's l1: 17.7197
[9000]  valid_set's l1: 17.642
[10000] valid_set's l1: 17.564
[1000]  valid_set's l1: 20.6342
[2000]  valid_set's l1: 19.7525
[3000]  valid_set's l1: 19.2481
[4000]  valid_set's l1: 18.9493
[5000]  valid_set's l1: 18.7222
[6000]  valid_set's l1: 18.5184
[7000]  valid_set's l1: 18.3862
[8000]  valid_set's l1: 18.2714
[9000]  valid_set's l1: 18.2004
[10000] valid_set's l1: 18.116
[1000]  valid_set's l1: 20.8518
[2000]  valid_set's l1: 20.0204
[3000]  valid_set's l1: 19.5508
[4000]  valid_set's l1: 19.2175
[5000]  valid_set's l1: 18.9815
[6000]  valid_set's l1: 18.8267
[7000]  valid_set's l1: 18.6892
[8000]  valid_set's l1: 18.569
[9000]  valid_set's l1: 18.4653
[10000] valid_set's l1: 18.3733
[1000]  valid_set's l1: 20.8399
[2000]  valid_set's l1: 19.926
[3000]  valid_set's l1: 19.39
[4000]  valid_set's l1: 19.0517
```

```
[5000]  valid_set's l1: 18.8307
[6000]  valid_set's l1: 18.6403
[7000]  valid_set's l1: 18.5231
[8000]  valid_set's l1: 18.4134
[9000]  valid_set's l1: 18.3341
[10000] valid_set's l1: 18.2712
[1000]  valid_set's l1: 21.0675
[2000]  valid_set's l1: 20.3308
[3000]  valid_set's l1: 19.8549
[4000]  valid_set's l1: 19.5564
[5000]  valid_set's l1: 19.3243
[6000]  valid_set's l1: 19.1303
[7000]  valid_set's l1: 18.9888
[8000]  valid_set's l1: 18.8919
[9000]  valid_set's l1: 18.8032
[10000] valid_set's l1: 18.7393
[1000]  valid_set's l1: 20.124
[2000]  valid_set's l1: 19.3629
[3000]  valid_set's l1: 18.891
[4000]  valid_set's l1: 18.5944
[5000]  valid_set's l1: 18.3539
[6000]  valid_set's l1: 18.2124
[7000]  valid_set's l1: 18.0752
[8000]  valid_set's l1: 17.9511
[9000]  valid_set's l1: 17.875
[10000] valid_set's l1: 17.7901
        -11.3667         = Validation score   (-mean_absolute_error)
        966.18s  = Training   runtime
        6.92s    = Validation runtime
Fitting model: WeightedEnsemble_L2 …
        -11.3313         = Validation score   (-mean_absolute_error)
        0.05s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 3758.97s … Best model:
"WeightedEnsemble_L2"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_133_C_seed_0/")
Beginning AutoGluon training …
AutoGluon will save models to "AutogluonModels/submission_133_C_seed_1/"
AutoGluon Version:  0.8.2
Python Version:     3.10.12
Operating System:   Darwin
Platform Machine:   arm64
Platform Version:   Darwin Kernel Version 22.1.0: Sun Oct  9 20:15:09 PDT 2022;
root:xnu-8792.41.9~2/RELEASE_ARM64_T6000
Disk Space Avail:   107.17 GB / 494.38 GB (21.7%)
Train Data Rows:    24416
```

```
Train Data Columns: 44
Tuning Data Rows:    1500
Tuning Data Columns: 44
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and label-values can't be converted to int).
        Label info (max, min, mean, stddev): (999.6, -0.0, 80.21597, 168.82736)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    4779.69 MB
        Train Data (Original)  Memory Usage: 10.42 MB (0.2% of available memory)
        Inferring data type of each feature based on column values. Set
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 2 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 3): ['elevation:m', 'snow_drift:idx',
'location']
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 40 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])   :  1 | ['is_estimated']
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 39 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', ['bool']) :  2 | ['snow_density:kgm3', 'is_estimated']
        0.1s = Fit runtime
        41 features in original data used to generate 41 features in processed
```

```
data.
        Train Data (Processed) Memory Usage: 8.14 MB (0.2% of available memory)
Data preprocessing and feature engineering runtime = 0.14s …
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
use_bag_holdout=True, will use tuning_data as holdout (will not be used for
early stopping).
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'XT': [{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}},
{'extra_trees': True, 'feature_fraction': 0.7832570544199176, 'learning_rate':
0.021720607471727896, 'min_data_in_leaf': 3, 'num_leaves': 21, 'ag_args':
{'name_suffix': '_r118', 'priority': 17}}],
        'FASTAI': {},
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
Excluded models: ['XT'] (Specified by `excluded_model_types`)
Fitting 6 L1 models …
Fitting model: KNeighborsUnif_BAG_L1 …

Training model for location C, seed 1…

        -19.3166         = Validation score   (-mean_absolute_error)
        0.02s    = Training   runtime
        315.0s   = Validation runtime
Fitting model: KNeighborsDist_BAG_L1 …
        -19.557  = Validation score   (-mean_absolute_error)
        0.02s    = Training   runtime
        312.12s  = Validation runtime
Fitting model: LightGBMXT_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy

[1000]  valid_set's l1: 19.2065
[2000]  valid_set's l1: 18.3505
[3000]  valid_set's l1: 18.0344
[4000]  valid_set's l1: 17.8789
[5000]  valid_set's l1: 17.8012
[6000]  valid_set's l1: 17.7661
[7000]  valid_set's l1: 17.7344
[8000]  valid_set's l1: 17.7234
[9000]  valid_set's l1: 17.7211
```

```
[10000] valid_set's l1: 17.7168
[1000]  valid_set's l1: 20.3328
[2000]  valid_set's l1: 19.6798
[3000]  valid_set's l1: 19.4441
[4000]  valid_set's l1: 19.2758
[5000]  valid_set's l1: 19.1633
[6000]  valid_set's l1: 19.047
[7000]  valid_set's l1: 18.9752
[8000]  valid_set's l1: 18.933
[9000]  valid_set's l1: 18.9004
[10000] valid_set's l1: 18.8821
[1000]  valid_set's l1: 19.193
[2000]  valid_set's l1: 18.6448
[3000]  valid_set's l1: 18.3984
[4000]  valid_set's l1: 18.2871
[5000]  valid_set's l1: 18.2058
[6000]  valid_set's l1: 18.1535
[7000]  valid_set's l1: 18.119
[8000]  valid_set's l1: 18.095
[9000]  valid_set's l1: 18.0767
[10000] valid_set's l1: 18.0705
[1000]  valid_set's l1: 19.8479
[2000]  valid_set's l1: 19.1082
[3000]  valid_set's l1: 18.7775
[4000]  valid_set's l1: 18.6305
[5000]  valid_set's l1: 18.5599
[6000]  valid_set's l1: 18.5088
[7000]  valid_set's l1: 18.4697
[8000]  valid_set's l1: 18.4507
[9000]  valid_set's l1: 18.4343
[10000] valid_set's l1: 18.4277
[1000]  valid_set's l1: 18.9725
[2000]  valid_set's l1: 18.5234
[3000]  valid_set's l1: 18.2881
[4000]  valid_set's l1: 18.1263
[5000]  valid_set's l1: 18.0636
[6000]  valid_set's l1: 17.995
[7000]  valid_set's l1: 17.9536
[8000]  valid_set's l1: 17.9134
[9000]  valid_set's l1: 17.8808
[10000] valid_set's l1: 17.8591
[1000]  valid_set's l1: 20.2082
[2000]  valid_set's l1: 19.5259
[3000]  valid_set's l1: 19.305
[4000]  valid_set's l1: 19.1277
[5000]  valid_set's l1: 19.0197
[6000]  valid_set's l1: 18.9373
[7000]  valid_set's l1: 18.8692
```

```
[8000]  valid_set's l1: 18.821
[9000]  valid_set's l1: 18.7784
[10000] valid_set's l1: 18.7605
[1000]  valid_set's l1: 18.8739
[2000]  valid_set's l1: 18.2251
[3000]  valid_set's l1: 17.8451
[4000]  valid_set's l1: 17.6694
[5000]  valid_set's l1: 17.5715
[6000]  valid_set's l1: 17.5098
[7000]  valid_set's l1: 17.4598
[8000]  valid_set's l1: 17.429
[9000]  valid_set's l1: 17.4069
[10000] valid_set's l1: 17.3924
[1000]  valid_set's l1: 19.7011
[2000]  valid_set's l1: 18.98
[3000]  valid_set's l1: 18.6952
[4000]  valid_set's l1: 18.5287
[5000]  valid_set's l1: 18.4333
[6000]  valid_set's l1: 18.3576
[7000]  valid_set's l1: 18.3092
[8000]  valid_set's l1: 18.2834
[9000]  valid_set's l1: 18.265
[10000] valid_set's l1: 18.2503
[1000]  valid_set's l1: 19.5234
[2000]  valid_set's l1: 18.9012
[3000]  valid_set's l1: 18.6105
[4000]  valid_set's l1: 18.4661
[5000]  valid_set's l1: 18.3722
[6000]  valid_set's l1: 18.3039
[7000]  valid_set's l1: 18.259
[8000]  valid_set's l1: 18.2328
[9000]  valid_set's l1: 18.212
[10000] valid_set's l1: 18.2065
[1000]  valid_set's l1: 19.6218
[2000]  valid_set's l1: 18.9713
[3000]  valid_set's l1: 18.7395
[4000]  valid_set's l1: 18.5443
[5000]  valid_set's l1: 18.4845
[6000]  valid_set's l1: 18.4199
[7000]  valid_set's l1: 18.3926
[8000]  valid_set's l1: 18.3714
[9000]  valid_set's l1: 18.3548
[10000] valid_set's l1: 18.3483
[1000]  valid_set's l1: 19.7096
[2000]  valid_set's l1: 19.0802
[3000]  valid_set's l1: 18.8222
[4000]  valid_set's l1: 18.7086
[5000]  valid_set's l1: 18.5713
```

```
[6000]  valid_set's l1: 18.4842
[7000]  valid_set's l1: 18.4197
[8000]  valid_set's l1: 18.3814
[9000]  valid_set's l1: 18.3542
[10000] valid_set's l1: 18.3267
[1000]  valid_set's l1: 19.5919
[2000]  valid_set's l1: 18.9941
[3000]  valid_set's l1: 18.624
[4000]  valid_set's l1: 18.3925
[5000]  valid_set's l1: 18.2561
[6000]  valid_set's l1: 18.1783
[7000]  valid_set's l1: 18.1137
[8000]  valid_set's l1: 18.0704
[9000]  valid_set's l1: 18.0456
[10000] valid_set's l1: 18.0241

        -11.3144         = Validation score   (-mean_absolute_error)
        1160.72s         = Training   runtime
        10.79s   = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -13.7311         = Validation score   (-mean_absolute_error)
        309.25s  = Training   runtime
        0.71s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -12.4453         = Validation score   (-mean_absolute_error)
        648.05s  = Training   runtime
        0.45s    = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy

[1000]  valid_set's l1: 20.4791
[2000]  valid_set's l1: 19.493
[3000]  valid_set's l1: 18.9549
[4000]  valid_set's l1: 18.5509
[5000]  valid_set's l1: 18.2954
[6000]  valid_set's l1: 18.083
[7000]  valid_set's l1: 17.9338
[8000]  valid_set's l1: 17.8053
[9000]  valid_set's l1: 17.7113
[10000] valid_set's l1: 17.6474
[1000]  valid_set's l1: 21.4538
[2000]  valid_set's l1: 20.5255
[3000]  valid_set's l1: 20.0158
[4000]  valid_set's l1: 19.6798
```

```
[5000]  valid_set's l1: 19.4049
[6000]  valid_set's l1: 19.2063
[7000]  valid_set's l1: 19.0522
[8000]  valid_set's l1: 18.9306
[9000]  valid_set's l1: 18.8158
[10000] valid_set's l1: 18.7231
[1000]  valid_set's l1: 19.9417
[2000]  valid_set's l1: 19.1113
[3000]  valid_set's l1: 18.7112
[4000]  valid_set's l1: 18.402
[5000]  valid_set's l1: 18.1735
[6000]  valid_set's l1: 18.0184
[7000]  valid_set's l1: 17.8958
[8000]  valid_set's l1: 17.7861
[9000]  valid_set's l1: 17.7083
[10000] valid_set's l1: 17.6393
[1000]  valid_set's l1: 20.7353
[2000]  valid_set's l1: 19.8554
[3000]  valid_set's l1: 19.3604
[4000]  valid_set's l1: 19.0192
[5000]  valid_set's l1: 18.775
[6000]  valid_set's l1: 18.5958
[7000]  valid_set's l1: 18.464
[8000]  valid_set's l1: 18.3502
[9000]  valid_set's l1: 18.2604
[10000] valid_set's l1: 18.1829
[1000]  valid_set's l1: 19.6932
[2000]  valid_set's l1: 19.0493
[3000]  valid_set's l1: 18.6227
[4000]  valid_set's l1: 18.3675
[5000]  valid_set's l1: 18.1757
[6000]  valid_set's l1: 17.9909
[7000]  valid_set's l1: 17.8669
[8000]  valid_set's l1: 17.7456
[9000]  valid_set's l1: 17.664
[10000] valid_set's l1: 17.5803
[1000]  valid_set's l1: 21.0761
[2000]  valid_set's l1: 20.3043
[3000]  valid_set's l1: 19.8535
[4000]  valid_set's l1: 19.5259
[5000]  valid_set's l1: 19.2921
[6000]  valid_set's l1: 19.1118
[7000]  valid_set's l1: 18.9655
[8000]  valid_set's l1: 18.8546
[9000]  valid_set's l1: 18.7469
[10000] valid_set's l1: 18.6826
[1000]  valid_set's l1: 20.0617
[2000]  valid_set's l1: 19.1158
```

```
[3000]  valid_set's l1: 18.5855
[4000]  valid_set's l1: 18.2925
[5000]  valid_set's l1: 18.0556
[6000]  valid_set's l1: 17.8468
[7000]  valid_set's l1: 17.6961
[8000]  valid_set's l1: 17.5711
[9000]  valid_set's l1: 17.4808
[10000] valid_set's l1: 17.3962
[1000]  valid_set's l1: 20.6915
[2000]  valid_set's l1: 19.7686
[3000]  valid_set's l1: 19.3032
[4000]  valid_set's l1: 18.9828
[5000]  valid_set's l1: 18.7433
[6000]  valid_set's l1: 18.5494
[7000]  valid_set's l1: 18.4258
[8000]  valid_set's l1: 18.3159
[9000]  valid_set's l1: 18.2253
[10000] valid_set's l1: 18.1493
[1000]  valid_set's l1: 20.6714
[2000]  valid_set's l1: 19.7586
[3000]  valid_set's l1: 19.3212
[4000]  valid_set's l1: 19.0476
[5000]  valid_set's l1: 18.8292
[6000]  valid_set's l1: 18.6349
[7000]  valid_set's l1: 18.4989
[8000]  valid_set's l1: 18.4082
[9000]  valid_set's l1: 18.316
[10000] valid_set's l1: 18.2635
[1000]  valid_set's l1: 20.8063
[2000]  valid_set's l1: 19.8705
[3000]  valid_set's l1: 19.3794
[4000]  valid_set's l1: 19.0029
[5000]  valid_set's l1: 18.7555
[6000]  valid_set's l1: 18.6117
[7000]  valid_set's l1: 18.4924
[8000]  valid_set's l1: 18.3765
[9000]  valid_set's l1: 18.299
[10000] valid_set's l1: 18.2346
[1000]  valid_set's l1: 20.8508
[2000]  valid_set's l1: 20.0051
[3000]  valid_set's l1: 19.556
[4000]  valid_set's l1: 19.1572
[5000]  valid_set's l1: 18.9124
[6000]  valid_set's l1: 18.7523
[7000]  valid_set's l1: 18.6116
[8000]  valid_set's l1: 18.5028
[9000]  valid_set's l1: 18.3984
[10000] valid_set's l1: 18.3244
```

```
[1000]  valid_set's l1: 20.373
[2000]  valid_set's l1: 19.6095
[3000]  valid_set's l1: 19.1625
[4000]  valid_set's l1: 18.8697
[5000]  valid_set's l1: 18.6507
[6000]  valid_set's l1: 18.4576
[7000]  valid_set's l1: 18.3182
[8000]  valid_set's l1: 18.2014
[9000]  valid_set's l1: 18.0889
[10000] valid_set's l1: 18.0108
        -11.0383         = Validation score   (-mean_absolute_error)
        923.33s  = Training   runtime
        6.59s    = Validation runtime
Fitting model: WeightedEnsemble_L2 …
        -10.8509         = Validation score   (-mean_absolute_error)
        0.06s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 3744.87s … Best model:
"WeightedEnsemble_L2"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_133_C_seed_1/")
Beginning AutoGluon training …
AutoGluon will save models to "AutogluonModels/submission_133_C_seed_2/"
AutoGluon Version:  0.8.2
Python Version:     3.10.12
Operating System:   Darwin
Platform Machine:   arm64
Platform Version:   Darwin Kernel Version 22.1.0: Sun Oct  9 20:15:09 PDT 2022;
root:xnu-8792.41.9~2/RELEASE_ARM64_T6000
Disk Space Avail:   106.46 GB / 494.38 GB (21.5%)
Train Data Rows:    24416
Train Data Columns: 44
Tuning Data Rows:    1500
Tuning Data Columns: 44
Label Column: y
Preprocessing data …
AutoGluon infers your prediction problem is: 'regression' (because dtype of
label-column == float and label-values can't be converted to int).
        Label info (max, min, mean, stddev): (999.6, -0.0, 80.39224, 169.07051)
        If 'regression' is not the correct problem_type, please manually specify
the problem_type parameter during predictor init (You may specify problem_type
as one of: ['binary', 'multiclass', 'regression'])
Using Feature Generators to preprocess the data …
Fitting AutoMLPipelineFeatureGenerator…
        Available Memory:                    4566.96 MB
        Train Data (Original)  Memory Usage: 10.42 MB (0.2% of available memory)
        Inferring data type of each feature based on column values. Set
```

```
feature_metadata_in to manually specify special dtypes of the features.
        Stage 1 Generators:
                Fitting AsTypeFeatureGenerator…
                        Note: Converting 2 features to boolean dtype as they
only contain 2 unique values.
        Stage 2 Generators:
                Fitting FillNaFeatureGenerator…
        Stage 3 Generators:
                Fitting IdentityFeatureGenerator…
        Stage 4 Generators:
                Fitting DropUniqueFeatureGenerator…
        Stage 5 Generators:
                Fitting DropDuplicatesFeatureGenerator…
        Useless Original Features (Count: 3): ['elevation:m', 'snow_drift:idx',
'location']
                These features carry no predictive signal and should be manually
investigated.
                This is typically a feature which has the same value for all
rows.
                These features do not need to be present at inference time.
        Types of features in original data (raw dtype, special dtypes):
                ('float', []) : 40 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', [])   :  1 | ['is_estimated']
        Types of features in processed data (raw dtype, special dtypes):
                ('float', [])     : 39 | ['absolute_humidity_2m:gm3',
'air_density_2m:kgm3', 'ceiling_height_agl:m', 'clear_sky_energy_1h:J',
'clear_sky_rad:W', …]
                ('int', ['bool']) :  2 | ['snow_density:kgm3', 'is_estimated']
        0.1s = Fit runtime
        41 features in original data used to generate 41 features in processed
data.
        Train Data (Processed) Memory Usage: 8.14 MB (0.2% of available memory)
Data preprocessing and feature engineering runtime = 0.11s …
AutoGluon will gauge predictive performance using evaluation metric:
'mean_absolute_error'
        This metric's sign has been flipped to adhere to being higher_is_better.
The metric score can be multiplied by -1 to get the metric value.
        To change this, specify the eval_metric parameter of Predictor()
use_bag_holdout=True, will use tuning_data as holdout (will not be used for
early stopping).
User-specified model hyperparameters to be fit:
{
        'NN_TORCH': {},
        'XT': [{'criterion': 'squared_error', 'ag_args': {'name_suffix': 'MSE',
'problem_types': ['regression', 'quantile']}}],
        'GBM': [{'extra_trees': True, 'ag_args': {'name_suffix': 'XT'}},
```

```
{'extra_trees': True, 'feature_fraction': 0.7832570544199176, 'learning_rate':
0.021720607471727896, 'min_data_in_leaf': 3, 'num_leaves': 21, 'ag_args':
{'name_suffix': '_r118', 'priority': 17}}],
        'FASTAI': {},
        'KNN': [{'weights': 'uniform', 'ag_args': {'name_suffix': 'Unif'}},
{'weights': 'distance', 'ag_args': {'name_suffix': 'Dist'}}],
}
Excluded models: ['XT'] (Specified by `excluded_model_types`)
Fitting 6 L1 models …
Fitting model: KNeighborsUnif_BAG_L1 …

Training model for location C, seed 2…

        -18.9068         = Validation score   (-mean_absolute_error)
        0.01s    = Training   runtime
        301.3s   = Validation runtime
Fitting model: KNeighborsDist_BAG_L1 …
        -19.1828         = Validation score   (-mean_absolute_error)
        0.02s    = Training   runtime
        308.0s   = Validation runtime
Fitting model: LightGBMXT_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy

[1000]  valid_set's l1: 19.3015
[2000]  valid_set's l1: 18.6328
[3000]  valid_set's l1: 18.2822
[4000]  valid_set's l1: 18.1193
[5000]  valid_set's l1: 18.0268
[6000]  valid_set's l1: 17.9528
[7000]  valid_set's l1: 17.9298
[8000]  valid_set's l1: 17.895
[9000]  valid_set's l1: 17.8804
[10000] valid_set's l1: 17.8699
[1000]  valid_set's l1: 20.2647
[2000]  valid_set's l1: 19.5117
[3000]  valid_set's l1: 19.1866
[4000]  valid_set's l1: 19.0033
[5000]  valid_set's l1: 18.9128
[6000]  valid_set's l1: 18.8348
[7000]  valid_set's l1: 18.7948
[8000]  valid_set's l1: 18.7591
[9000]  valid_set's l1: 18.7249
[10000] valid_set's l1: 18.7049
[1000]  valid_set's l1: 18.8155
[2000]  valid_set's l1: 18.2204
[3000]  valid_set's l1: 17.9606
[4000]  valid_set's l1: 17.8024
[5000]  valid_set's l1: 17.7265
```

```
[6000]  valid_set's l1: 17.6356
[7000]  valid_set's l1: 17.6081
[8000]  valid_set's l1: 17.5858
[9000]  valid_set's l1: 17.5645
[10000] valid_set's l1: 17.5478
[1000]  valid_set's l1: 19.6054
[2000]  valid_set's l1: 18.9729
[3000]  valid_set's l1: 18.6702
[4000]  valid_set's l1: 18.4985
[5000]  valid_set's l1: 18.4176
[6000]  valid_set's l1: 18.3725
[7000]  valid_set's l1: 18.3357
[8000]  valid_set's l1: 18.3141
[9000]  valid_set's l1: 18.2924
[10000] valid_set's l1: 18.2807
[1000]  valid_set's l1: 19.3067
[2000]  valid_set's l1: 18.7261
[3000]  valid_set's l1: 18.3898
[4000]  valid_set's l1: 18.2543
[5000]  valid_set's l1: 18.1439
[6000]  valid_set's l1: 18.1008
[7000]  valid_set's l1: 18.0436
[8000]  valid_set's l1: 18.0138
[9000]  valid_set's l1: 17.9923
[10000] valid_set's l1: 17.9636
[1000]  valid_set's l1: 20.2196
[2000]  valid_set's l1: 19.5949
[3000]  valid_set's l1: 19.3329
[4000]  valid_set's l1: 19.1443
[5000]  valid_set's l1: 19.0406
[6000]  valid_set's l1: 18.9613
[7000]  valid_set's l1: 18.8997
[8000]  valid_set's l1: 18.87
[9000]  valid_set's l1: 18.8264
[10000] valid_set's l1: 18.8078
[1000]  valid_set's l1: 18.8101
[2000]  valid_set's l1: 18.1497
[3000]  valid_set's l1: 17.858
[4000]  valid_set's l1: 17.7096
[5000]  valid_set's l1: 17.6137
[6000]  valid_set's l1: 17.5533
[7000]  valid_set's l1: 17.5054
[8000]  valid_set's l1: 17.4727
[9000]  valid_set's l1: 17.4551
[10000] valid_set's l1: 17.4351
[1000]  valid_set's l1: 19.7588
[2000]  valid_set's l1: 19.1403
[3000]  valid_set's l1: 18.8211
```

```
[4000]  valid_set's l1: 18.664
[5000]  valid_set's l1: 18.5666
[6000]  valid_set's l1: 18.4997
[7000]  valid_set's l1: 18.4456
[8000]  valid_set's l1: 18.41
[9000]  valid_set's l1: 18.3813
[10000] valid_set's l1: 18.3536
[1000]  valid_set's l1: 19.8098
[2000]  valid_set's l1: 19.0578
[3000]  valid_set's l1: 18.7209
[4000]  valid_set's l1: 18.5494
[5000]  valid_set's l1: 18.4274
[6000]  valid_set's l1: 18.3437
[7000]  valid_set's l1: 18.2823
[8000]  valid_set's l1: 18.2463
[9000]  valid_set's l1: 18.2187
[10000] valid_set's l1: 18.2029
[1000]  valid_set's l1: 19.8458
[2000]  valid_set's l1: 19.1899
[3000]  valid_set's l1: 18.8619
[4000]  valid_set's l1: 18.6872
[5000]  valid_set's l1: 18.596
[6000]  valid_set's l1: 18.5371
[7000]  valid_set's l1: 18.5102
[8000]  valid_set's l1: 18.4904
[9000]  valid_set's l1: 18.4724
[10000] valid_set's l1: 18.4628
[1000]  valid_set's l1: 19.8736
[2000]  valid_set's l1: 19.2015
[3000]  valid_set's l1: 18.9176
[4000]  valid_set's l1: 18.77
[5000]  valid_set's l1: 18.6679
[6000]  valid_set's l1: 18.6093
[7000]  valid_set's l1: 18.5743
[8000]  valid_set's l1: 18.5387
[9000]  valid_set's l1: 18.508
[10000] valid_set's l1: 18.4947
[1000]  valid_set's l1: 19.6707
[2000]  valid_set's l1: 19.081
[3000]  valid_set's l1: 18.7355
[4000]  valid_set's l1: 18.5551
[5000]  valid_set's l1: 18.4539
[6000]  valid_set's l1: 18.3848
[7000]  valid_set's l1: 18.3106
[8000]  valid_set's l1: 18.2699
[9000]  valid_set's l1: 18.2353
[10000] valid_set's l1: 18.2173
```

```
        -10.1918         = Validation score   (-mean_absolute_error)
        1115.53s         = Training   runtime
        11.37s   = Validation runtime
Fitting model: NeuralNetFastAI_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -12.4923         = Validation score   (-mean_absolute_error)
        234.19s  = Training   runtime
        0.42s    = Validation runtime
Fitting model: NeuralNetTorch_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy
        -12.6592         = Validation score   (-mean_absolute_error)
        452.06s  = Training   runtime
        0.43s    = Validation runtime
Fitting model: LightGBM_r118_BAG_L1 …
        Fitting 12 child models (S1F1 - S2F6) | Fitting with
SequentialLocalFoldFittingStrategy

[1000]  valid_set's l1: 20.492
[2000]  valid_set's l1: 19.5621
[3000]  valid_set's l1: 19.0414
[4000]  valid_set's l1: 18.6862
[5000]  valid_set's l1: 18.4338
[6000]  valid_set's l1: 18.2248
[7000]  valid_set's l1: 18.0276
[8000]  valid_set's l1: 17.9057
[9000]  valid_set's l1: 17.8047
[10000] valid_set's l1: 17.723
[1000]  valid_set's l1: 21.3125
[2000]  valid_set's l1: 20.3927
[3000]  valid_set's l1: 19.9035
[4000]  valid_set's l1: 19.5618
[5000]  valid_set's l1: 19.2808
[6000]  valid_set's l1: 19.0691
[7000]  valid_set's l1: 18.9404
[8000]  valid_set's l1: 18.8347
[9000]  valid_set's l1: 18.7269
[10000] valid_set's l1: 18.6491
[1000]  valid_set's l1: 19.6474
[2000]  valid_set's l1: 18.8483
[3000]  valid_set's l1: 18.3734
[4000]  valid_set's l1: 18.1017
[5000]  valid_set's l1: 17.8729
[6000]  valid_set's l1: 17.7036
[7000]  valid_set's l1: 17.5607
[8000]  valid_set's l1: 17.4558
[9000]  valid_set's l1: 17.36
```

```
[10000] valid_set's l1: 17.2824
[1000]  valid_set's l1: 20.8646
[2000]  valid_set's l1: 19.9455
[3000]  valid_set's l1: 19.4873
[4000]  valid_set's l1: 19.2384
[5000]  valid_set's l1: 18.9751
[6000]  valid_set's l1: 18.7786
[7000]  valid_set's l1: 18.6455
[8000]  valid_set's l1: 18.5422
[9000]  valid_set's l1: 18.4609
[10000] valid_set's l1: 18.3823
[1000]  valid_set's l1: 19.8491
[2000]  valid_set's l1: 19.2448
[3000]  valid_set's l1: 18.8393
[4000]  valid_set's l1: 18.5061
[5000]  valid_set's l1: 18.3142
[6000]  valid_set's l1: 18.1408
[7000]  valid_set's l1: 18.0198
[8000]  valid_set's l1: 17.9032
[9000]  valid_set's l1: 17.8107
[10000] valid_set's l1: 17.7466
[1000]  valid_set's l1: 21.1663
[2000]  valid_set's l1: 20.3867
[3000]  valid_set's l1: 20.004
[4000]  valid_set's l1: 19.7141
[5000]  valid_set's l1: 19.5323
[6000]  valid_set's l1: 19.3595
[7000]  valid_set's l1: 19.2365
[8000]  valid_set's l1: 19.1316
[9000]  valid_set's l1: 19.056
[10000] valid_set's l1: 19.0019
[1000]  valid_set's l1: 19.8712
[2000]  valid_set's l1: 19.0134
[3000]  valid_set's l1: 18.5051
[4000]  valid_set's l1: 18.1851
[5000]  valid_set's l1: 17.9249
[6000]  valid_set's l1: 17.7183
[7000]  valid_set's l1: 17.588
[8000]  valid_set's l1: 17.4523
[9000]  valid_set's l1: 17.3276
[10000] valid_set's l1: 17.2502
[1000]  valid_set's l1: 20.5829
[2000]  valid_set's l1: 19.7017
[3000]  valid_set's l1: 19.2151
[4000]  valid_set's l1: 18.8704
[5000]  valid_set's l1: 18.6479
[6000]  valid_set's l1: 18.473
[7000]  valid_set's l1: 18.3316
```

```
[8000]  valid_set's l1: 18.2273
[9000]  valid_set's l1: 18.1184
[10000] valid_set's l1: 18.0431
[1000]  valid_set's l1: 20.8973
[2000]  valid_set's l1: 20.0003
[3000]  valid_set's l1: 19.5236
[4000]  valid_set's l1: 19.1702
[5000]  valid_set's l1: 18.9642
[6000]  valid_set's l1: 18.7936
[7000]  valid_set's l1: 18.6617
[8000]  valid_set's l1: 18.5584
[9000]  valid_set's l1: 18.4618
[10000] valid_set's l1: 18.3808
[1000]  valid_set's l1: 20.8319
[2000]  valid_set's l1: 19.9435
[3000]  valid_set's l1: 19.4006
[4000]  valid_set's l1: 19.051
[5000]  valid_set's l1: 18.8388
[6000]  valid_set's l1: 18.6639
[7000]  valid_set's l1: 18.5457
[8000]  valid_set's l1: 18.4612
[9000]  valid_set's l1: 18.3736
[10000] valid_set's l1: 18.3259
[1000]  valid_set's l1: 20.7456
[2000]  valid_set's l1: 19.9707
[3000]  valid_set's l1: 19.4987
[4000]  valid_set's l1: 19.1251
[5000]  valid_set's l1: 18.8626
[6000]  valid_set's l1: 18.6797
[7000]  valid_set's l1: 18.531
[8000]  valid_set's l1: 18.4044
[9000]  valid_set's l1: 18.32
[10000] valid_set's l1: 18.24
[1000]  valid_set's l1: 20.3648
[2000]  valid_set's l1: 19.6182
[3000]  valid_set's l1: 19.0977
[4000]  valid_set's l1: 18.7807
[5000]  valid_set's l1: 18.5573
[6000]  valid_set's l1: 18.3851
[7000]  valid_set's l1: 18.254
[8000]  valid_set's l1: 18.1501
[9000]  valid_set's l1: 18.057
[10000] valid_set's l1: 17.9757
        -10.0536         = Validation score   (-mean_absolute_error)
        957.07s  = Training   runtime
        5.69s    = Validation runtime
Fitting model: WeightedEnsemble_L2 …
```

```
        -10.0073        = Validation score   (-mean_absolute_error)
        0.06s    = Training   runtime
        0.0s     = Validation runtime
AutoGluon training complete, total runtime = 3436.99s … Best model:
"WeightedEnsemble_L2"
TabularPredictor saved. To load, use: predictor =
TabularPredictor.load("AutogluonModels/submission_133_C_seed_2/")
```

[29]:
```
# analyse weights in ensemble
for i in range(len(predictors)):
    for j in range(len(predictors[i].predictors)):
        print(f"Predictor {i}, seed {j}:")
        print(predictors[i].predictors[j].
 ↪info()["model_info"]["WeightedEnsemble_L2"]["children_info"]["S1F1"]["model_weights"])
```

```
Predictor 0, seed 0:
{'LightGBMXT_BAG_L1': 0.32142857142857145, 'NeuralNetTorch_BAG_L1':
0.5535714285714286, 'LightGBM_r118_BAG_L1': 0.125}
Predictor 0, seed 1:
{'LightGBMXT_BAG_L1': 0.5074626865671642, 'NeuralNetTorch_BAG_L1':
0.417910447761194, 'LightGBM_r118_BAG_L1': 0.07462686567164178}
Predictor 0, seed 2:
{'LightGBMXT_BAG_L1': 0.7407407407407407, 'NeuralNetFastAI_BAG_L1':
0.024691358024691357, 'NeuralNetTorch_BAG_L1': 0.09876543209876543,
'LightGBM_r118_BAG_L1': 0.13580246913580246}
Predictor 1, seed 0:
{'LightGBMXT_BAG_L1': 0.25925925925925924, 'NeuralNetFastAI_BAG_L1':
0.037037037037037035, 'NeuralNetTorch_BAG_L1': 0.3888888888888889,
'LightGBM_r118_BAG_L1': 0.3148148148148148}
Predictor 1, seed 1:
{'LightGBMXT_BAG_L1': 0.39759036144578314, 'NeuralNetFastAI_BAG_L1':
0.03614457831325301, 'NeuralNetTorch_BAG_L1': 0.4939759036144578,
'LightGBM_r118_BAG_L1': 0.07228915662650602}
Predictor 1, seed 2:
{'LightGBMXT_BAG_L1': 0.2828282828282828, 'NeuralNetFastAI_BAG_L1':
0.12121212121212122, 'NeuralNetTorch_BAG_L1': 0.3434343434343434,
'LightGBM_r118_BAG_L1': 0.25252525252525254}
Predictor 2, seed 0:
{'LightGBMXT_BAG_L1': 0.23376623376623376, 'NeuralNetFastAI_BAG_L1':
0.025974025974025976, 'NeuralNetTorch_BAG_L1': 0.07792207792207792,
'LightGBM_r118_BAG_L1': 0.6623376623376623}
Predictor 2, seed 1:
{'KNeighborsDist_BAG_L1': 0.0547945205479452, 'LightGBMXT_BAG_L1':
0.2465753424657534, 'NeuralNetTorch_BAG_L1': 0.2191780821917808,
'LightGBM_r118_BAG_L1': 0.4794520547945205}
Predictor 2, seed 2:
{'KNeighborsUnif_BAG_L1': 0.01, 'LightGBMXT_BAG_L1': 0.31,
'LightGBM_r118_BAG_L1': 0.68}
```

4 Submit

[30]:
```
import pandas as pd
import matplotlib.pyplot as plt

future_test_data = TabularDataset('X_test_raw.csv')
future_test_data["ds"] = pd.to_datetime(future_test_data["ds"])
#test_data
```

```
Loaded data from: X_test_raw.csv | Columns = 45 / 45 | Rows = 4608 -> 4608
```

[31]:
```
test_ids = TabularDataset('test.csv')
test_ids["time"] = pd.to_datetime(test_ids["time"])
# merge test_data with test_ids
future_test_data_merged = pd.merge(future_test_data, test_ids, how="inner",␣
 ↪right_on=["time", "location"], left_on=["ds", "location"])

#test_data_merged
```

```
Loaded data from: test.csv | Columns = 4 / 4 | Rows = 2160 -> 2160
```

[54]:
```
# predict, grouped by location
predictions = []
location_map = {
    "A": 0,
    "B": 1,
    "C": 2
}
for loc, group in future_test_data.groupby('location'):
    i = location_map[loc]
    subset = future_test_data_merged[future_test_data_merged["location"] ==␣
 ↪loc].reset_index(drop=True)
    #print(subset)
    pred = predictors[i].predict(subset)
    subset["prediction"] = pred * 1.018 if loc=="A" else pred
    predictions.append(subset)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/Users/jorgensandhaug/Desktop/tdt4173/TDT4173/short_2.ipynb Cell 26 line 1

     <a href='vscode-notebook-cell:/Users/jorgensandhaug/Desktop/tdt4173/TDT417
 ↪short_2.ipynb#X45sZmlsZQ%3D%3D?line=9'>10</a> subset =␣
 ↪future_test_data_merged[future_test_data_merged["location"] == loc].
 ↪reset_index(drop=True)
     <a href='vscode-notebook-cell:/Users/jorgensandhaug/Desktop/tdt4173/TDT417
 ↪short_2.ipynb#X45sZmlsZQ%3D%3D?line=10'>11</a> #print(subset)
---> <a href='vscode-notebook-cell:/Users/jorgensandhaug/Desktop/tdt4173/TDT417
 ↪short_2.ipynb#X45sZmlsZQ%3D%3D?line=11'>12</a> pred = predictors[i].
 ↪predict(subset)
```

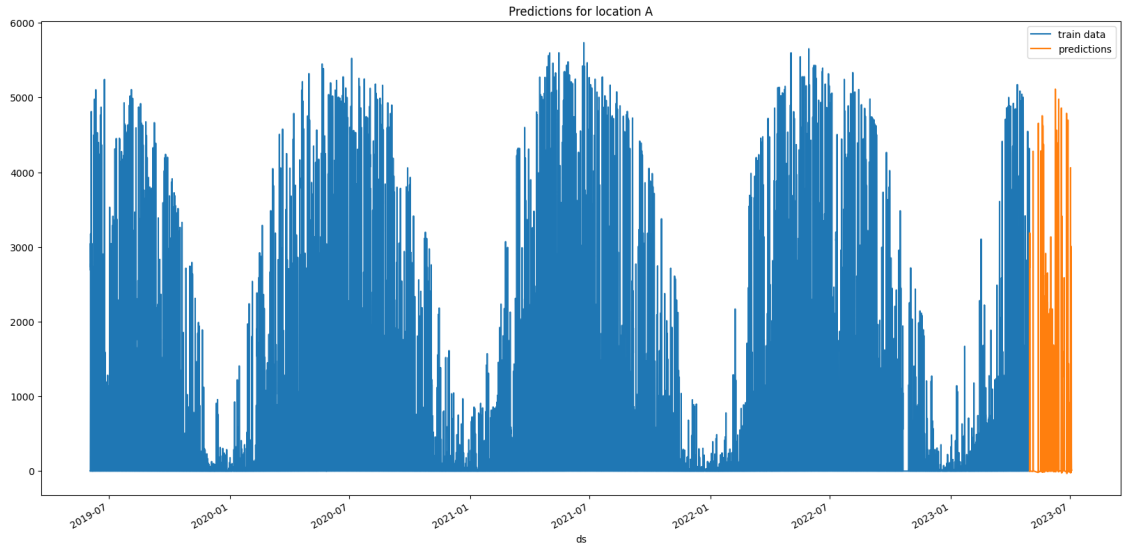
```
     <a href='vscode-notebook-cell:/Users/jorgensandhaug/Desktop/tdt4173/TDT4173/
 ↪short_2.ipynb#X45sZmlsZQ%3D%3D?line=12'>13</a> subset["prediction"] = pred
     <a href='vscode-notebook-cell:/Users/jorgensandhaug/Desktop/tdt4173/TDT4173/
 ↪short_2.ipynb#X45sZmlsZQ%3D%3D?line=13'>14</a> predictions.append(subset)

/Users/jorgensandhaug/Desktop/tdt4173/TDT4173/short_2.ipynb Cell 26 line 9

      <a href='vscode-notebook-cell:/Users/jorgensandhaug/Desktop/tdt4173/
 ↪TDT4173/short_2.ipynb#X45sZmlsZQ%3D%3D?line=6'>7</a> predictions = []
      <a href='vscode-notebook-cell:/Users/jorgensandhaug/Desktop/tdt4173/
 ↪TDT4173/short_2.ipynb#X45sZmlsZQ%3D%3D?line=7'>8</a> for predictor in self.
 ↪predictors:
----> <a href='vscode-notebook-cell:/Users/jorgensandhaug/Desktop/tdt4173/
 ↪TDT4173/short_2.ipynb#X45sZmlsZQ%3D%3D?line=8'>9</a>     predictions.
 ↪append(predictor.predict(x))
     <a href='vscode-notebook-cell:/Users/jorgensandhaug/Desktop/tdt4173/TDT4173/
 ↪short_2.ipynb#X45sZmlsZQ%3D%3D?line=9'>10</a> return np.mean(predictions,␣
 ↪axis=0)

File /opt/homebrew/anaconda3/envs/ag/lib/python3.10/site-packages/autogluon/
 ↪tabular/predictor/predictor.py:1572, in TabularPredictor.predict(self, data,␣
 ↪model, as_pandas, transform_features, decision_threshold)
   1570 if decision_threshold is None:
   1571     decision_threshold = self.decision_threshold
-> 1572 return self._learner.predict(X=data, model=model, as_pandas=as_pandas,␣
 ↪transform_features=transform_features, decision_threshold=decision_threshold)

File /opt/homebrew/anaconda3/envs/ag/lib/python3.10/site-packages/autogluon/
 ↪tabular/learner/abstract_learner.py:208, in AbstractTabularLearner.
 ↪predict(self, X, model, as_pandas, inverse_transform, transform_features,␣
 ↪decision_threshold)
    206     decision_threshold = 0.5
    207 X_index = copy.deepcopy(X.index) if as_pandas else None
--> 208 y_pred_proba = self.predict_proba(
    209     X=X, model=model, as_pandas=False, as_multiclass=False,␣
 ↪inverse_transform=False, transform_features=transform_features
    210 )
    211 problem_type = self.label_cleaner.problem_type_transform or self.
 ↪problem_type
    212 y_pred = get_pred_from_proba(y_pred_proba=y_pred_proba,␣
 ↪problem_type=problem_type, decision_threshold=decision_threshold)

File /opt/homebrew/anaconda3/envs/ag/lib/python3.10/site-packages/autogluon/
 ↪tabular/learner/abstract_learner.py:189, in AbstractTabularLearner.
 ↪predict_proba(self, X, model, as_pandas, as_multiclass, inverse_transform,␣
 ↪transform_features)
    187     if transform_features:
    188         X = self.transform_features(X)
--> 189     y_pred_proba = self.load_trainer().predict_proba(X, model=model)
    190 y_pred_proba = self._post_process_predict_proba(
```

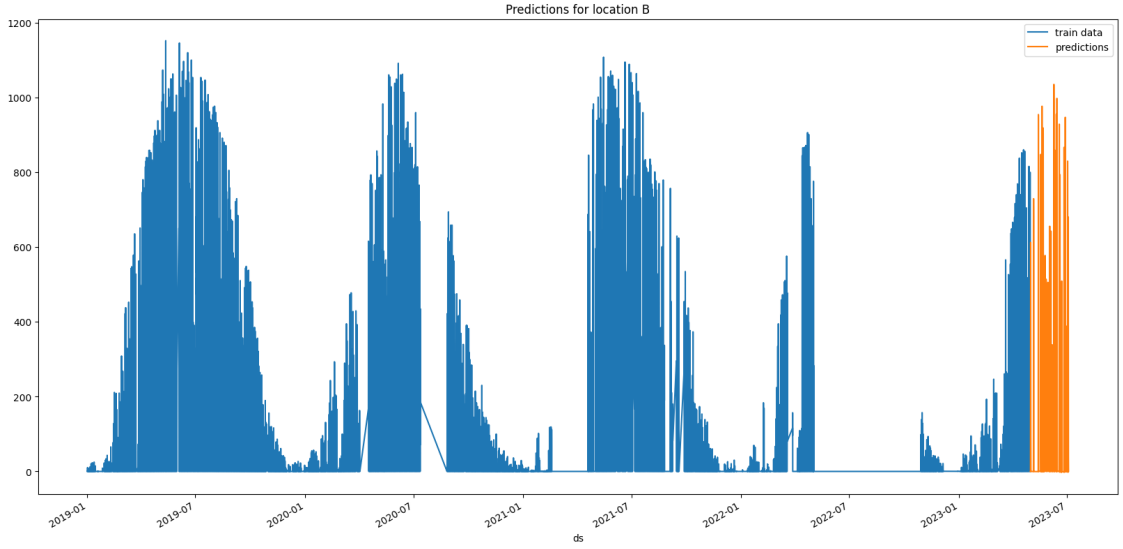
```
    191     y_pred_proba=y_pred_proba, as_pandas=as_pandas, index=X_index,␣
 ↪as_multiclass=as_multiclass, inverse_transform=inverse_transform
    192 )
    193 return y_pred_proba

File /opt/homebrew/anaconda3/envs/ag/lib/python3.10/site-packages/autogluon/cor /
 ↪trainer/abstract_trainer.py:743, in AbstractTrainer.predict_proba(self, X,␣
 ↪model)
    741     model = self._get_best()
    742 cascade = isinstance(model, list)
--> 743 return self._predict_proba_model(X, model, cascade=cascade)

File /opt/homebrew/anaconda3/envs/ag/lib/python3.10/site-packages/autogluon/cor /
 ↪trainer/abstract_trainer.py:2440, in AbstractTrainer.
 ↪_predict_proba_model(self, X, model, model_pred_proba_dict, cascade)
   2439 def _predict_proba_model(self, X, model, model_pred_proba_dict=None,␣
 ↪cascade=False):
-> 2440     return self.get_pred_proba_from_model(model=model, X=X,␣
 ↪model_pred_proba_dict=model_pred_proba_dict, cascade=cascade)

File /opt/homebrew/anaconda3/envs/ag/lib/python3.10/site-packages/autogluon/cor /
 ↪trainer/abstract_trainer.py:757, in AbstractTrainer.
 ↪get_pred_proba_from_model(self, model, X, model_pred_proba_dict, cascade)
    755 else:
    756     models = [model]
--> 757 model_pred_proba_dict = self.get_model_pred_proba_dict(X=X,␣
 ↪models=models, model_pred_proba_dict=model_pred_proba_dict, cascade=cascade)
    758 if not isinstance(model, str):
    759     model = model.name

File /opt/homebrew/anaconda3/envs/ag/lib/python3.10/site-packages/autogluon/cor /
 ↪trainer/abstract_trainer.py:1006, in AbstractTrainer.
 ↪get_model_pred_proba_dict(self, X, models, model_pred_proba_dict,␣
 ↪model_pred_time_dict, record_pred_time, use_val_cache, cascade,␣
 ↪cascade_threshold)
   1004     else:
   1005         preprocess_kwargs = dict(infer=False,␣
 ↪model_pred_proba_dict=model_pred_proba_dict)
-> 1006     model_pred_proba_dict[model_name] = model.predict_proba(X,␣
 ↪**preprocess_kwargs)
   1007 else:
   1008     model_pred_proba_dict[model_name] = model.predict_proba(X)

File /opt/homebrew/anaconda3/envs/ag/lib/python3.10/site-packages/autogluon/cor /
 ↪models/ensemble/bagged_ensemble_model.py:348, in BaggedEnsembleModel.
 ↪predict_proba(self, X, normalize, **kwargs)
    346 pred_proba = model.predict_proba(X=X, preprocess_nonadaptive=False,␣
 ↪normalize=normalize)
    347 for model in self.models[1:]:
```

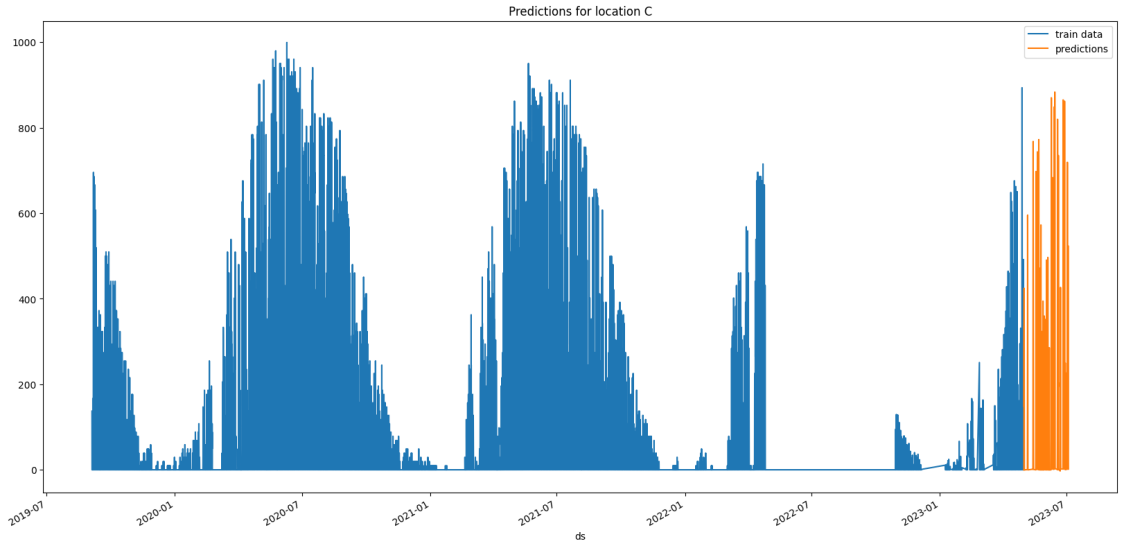
```
--> 348     model = self.load_child(model)
    349     pred_proba += model.predict_proba(X=X, preprocess_nonadaptive=False␣
 ↪normalize=normalize)
    350 pred_proba = pred_proba / len(self.models)

File /opt/homebrew/anaconda3/envs/ag/lib/python3.10/site-packages/autogluon/cor /
 ↪models/ensemble/bagged_ensemble_model.py:752, in BaggedEnsembleModel.
 ↪load_child(self, model, verbose)
    750 if isinstance(model, str):
    751     child_path = self.create_contexts(os.path.join(self.path, model))
--> 752     return self._child_type.load(path=child_path, verbose=verbose)
    753 else:
    754     return model

File /opt/homebrew/anaconda3/envs/ag/lib/python3.10/site-packages/autogluon/cor /
 ↪models/abstract/abstract_model.py:1063, in AbstractModel.load(cls, path,␣
 ↪reset_paths, verbose)
   1041 """
   1042 Loads the model from disk to memory.
   1043
   (…)
   1060     Loaded model object.
   1061 """
   1062 file_path = os.path.join(path, cls.model_file_name)
-> 1063 model = load_pkl.load(path=file_path, verbose=verbose)
   1064 if reset_paths:
   1065     model.set_contexts(path)

File /opt/homebrew/anaconda3/envs/ag/lib/python3.10/site-packages/autogluon/
 ↪common/loaders/load_pkl.py:44, in load(path, format, verbose, **kwargs)
     42 if compression_fn in compression_fn_map:
     43     with compression_fn_map[compression_fn]["open"](validated_path,␣
 ↪"rb", **compression_fn_kwargs) as fin:
---> 44         object = pickle.load(fin)
     45 else:
     46     raise ValueError(
     47         f"compression_fn={compression_fn} or␣
 ↪compression_fn_kwargs={compression_fn_kwargs} are not valid."
     48         f" Valid function values: {compression_fn_map.keys()}"
     49     )

File /opt/homebrew/anaconda3/envs/ag/lib/python3.10/site-packages/lightgbm/basi .
 ↪py:2690, in Booster.__setstate__(self, state)
   2688     handle = ctypes.c_void_p()
   2689     out_num_iterations = ctypes.c_int(0)
-> 2690     _safe_call(_LIB.LGBM_BoosterLoadModelFromString(
   2691         c_str(model_str),
   2692         ctypes.byref(out_num_iterations),
```

```
   2693         ctypes.byref(handle)))
   2694     state['handle'] = handle
   2695 self.__dict__.update(state)

KeyboardInterrupt:
```

[ ]:
```
for loc, idx in location_map.items():
    fig, ax = plt.subplots(figsize=(20, 10))
    # plot train data
    train_data_list[0][train_data_list[0]["location"]==loc].plot(x='ds', y='y',␣
 ↪ax=ax, label="train data")
    # plot predictions
    predictions[idx].plot(x='ds', y='prediction', ax=ax, label="predictions")

    ax.set_title(f"Predictions for location {loc}")
```







[ ]:
```python
temp_predictions = [prediction.copy() for prediction in predictions]
if clip_predictions:
    # clip predictions smaller than 0 to 0
    for pred in temp_predictions:
        # print smallest prediction
        print("Smallest prediction:", pred["prediction"].min())
        pred.loc[pred["prediction"] < 0, "prediction"] = 0
        print("Smallest prediction after clipping:", pred["prediction"].min())
```

```
# concatenate predictions
submissions_df = pd.concat(temp_predictions)
submissions_df = submissions_df[["id", "prediction"]]
submissions_df
```

```
Smallest prediction: -33.07142
Smallest prediction after clipping: 0.0
Smallest prediction: -2.5387175
Smallest prediction after clipping: 0.0
Smallest prediction: -2.7704537
Smallest prediction after clipping: 0.0
```

[ ]:
```
        id  prediction
0        0    0.000000
1        1    0.000000
2        2    0.000000
3        3   23.330450
4        4  327.811432
..     …          …
715   2155   69.809311
716   2156   38.407642
717   2157   11.460097
718   2158    2.336670
719   2159    1.775181

[2160 rows x 2 columns]
```

[ ]:
```
# Save the submission
print(f"Saving submission to submissions/{new_filename}.csv")
submissions_df.to_csv(os.path.join('submissions', f"{new_filename}.csv"),␣
 ↪index=False)
```

```
Saving submission to submissions/submission_133.csv
```

[ ]:
```
# save this notebook to submissions folder
import subprocess
import os
subprocess.run(["jupyter", "nbconvert", "--to", "pdf", "--output", os.path.
 ↪join('notebook_pdfs', f"{new_filename}.pdf"), "short_2.ipynb"])
```

```
[NbConvertApp] Converting notebook short_2.ipynb to pdf
[NbConvertApp] Support files will be in notebook_pdfs/submission_133_files/
[NbConvertApp] Making directory
./notebook_pdfs/submission_133_files/notebook_pdfs
[NbConvertApp] Writing 232653 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
```

```
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 363275 bytes to notebook_pdfs/submission_133.pdf
```

[ ]:
```
CompletedProcess(args=['jupyter', 'nbconvert', '--to', 'pdf', '--output',
'notebook_pdfs/submission_133.pdf', 'short_2.ipynb'], returncode=0)
```